```
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        
```

```
  SSSSSSSS  HH      HH  WW      WW    CCCCCCCC  LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
  SSSSSSSS  HH      HH  WW      WW    CCCCCCCC  LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
  SSSSSS    HHHHHHHHHH  WW      WW  CC          LL            SSSSSS        TT      RRRRRRRR
  SSSSSS    HHHHHHHHHH  WW      WW  CC          LL            SSSSSS        TT      RRRRRRRR
        SS  HH      HH  WW  WW  WW  CC          LL                  SS      TT      RR  RR
        SS  HH      HH  WW  WW  WW  CC          LL                  SS      TT      RR  RR
        SS  HH      HH  WWWW  WWWW  CC          LL                  SS      TT      RR    RR        ....
        SS  HH      HH  WWWW  WWWW  CC          LL                  SS      TT      RR    RR        ....
SSSSSSSS    HH      HH  WW      WW    CCCCCCCC  LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....
SSSSSSSS    HH      HH  WW      WW    CCCCCCCC  LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(1) 3 Copyright notice
(2) 29 Program description
(3) 162 Definitions.
(3) 664 Ascii strings.
(4) 1146 Local storage.
(6) 1873 Macro definitions.
(6) 1874 - Conditional Branch Macro.
(7) 2100 - DEFINE_FIELD macro.
(8) 2460 Cluster report field descriptors.
(9) 2670 Local status report field descriptors.
(10) 2783 Main program loop.
(19) 3545 Initialization.
(21) 4086 Exit handler.
(24) 4399 SCS report generation routine.
(29) 4822 Local report generation routine.
(30) 4897 Error message output.
(31) 4984 Broadcast message handling.
(32) 5067 Scrolling routines.
(32) 5068 - SET_SCROLL routine.
(33) 5265 - CHECK_MORE routine.
(34) 5357 - Set scrolling region routines.
(35) 5420 AST routines.
(39) 5681 Display header generation.
(40) 5822 - Output the display description
(41) 5905 - Output the time
(42) 6006 - Box drawing routines.
(44) 6226 - DRAW_LAST_LINE
(45) 6290 - Field title generation.
(47) 6503 - Class title generation.
(48) 6606 REFRESH routine.
(49) 6646 FIND_ENTRY routine.
(50) 6736 Miscellaneous routines.
(58) 7134 Data formatting routines.
(63) 7519 Data display routines.
(64) 7547 - GET_DATA routine.
(65) 7784 - General data display routines.
(67) 7927 - Cluster class display routines.
(68) 8015 - Systems class display routines.
(69) 8076 - Cluster system block display routines.
(74) 8245 - Circuits class display routines.
(79) 8672 - Local port display routines.
(84) 8924 - CHECK_DIF routine.
(85) 9038 Description of snapshot buffers.
(86) 9302 Snapshot buffers.
(87) 9372 Snapshot routines.
(89) 10322 - Interlock Queue routine.
(91) 10418 - GET_NUM_PB
(92) 10480 Parser State Table.
(110) 11421 Tparse action routines.
(110) 11422 - SHOW_CLUSTER
(111) 11494 - SHOW_LOCAL
(112) 11562 - ADD_FIELD
(113) 11659 - REMOVE_FIELD
(114) 11738 - LOOKUP_FIELD
(115) 11832 - ADD_ALL_FIELDS
(116) 11898 - INCLUDE_CLASS
(117) 11958 - EXCLUDE_CLASS
(118) 12019 - ADD_ID
(119) 12067 - REM_NODE
(120) 12119 - REM_ID

```
(121)   12169        -       ADD_NODE
(122)   12222        -       FIND_NODE
(123)   12294        -       ADD_TO_LIST
(124)   12376        -       REMOVE_FROM_LIST
(125)   12432        -       FIND_SYS_ID
(126)   12496        -       CONVERT_TO_HEX
(127)   12602        -       SET_WIDTH
(128)   12662        -       SET_FORMAT
(129)   12762        -       SET_INTERVAL
(130)   12802        -       SET_SCREEN
(131)   12895        -       GET_HELP
(132)   12980        -       RESET
(133)   13107        -       EXECUTE
```

```
0000     1          .TITLE  SHWCLSTR       Show Cluster Utility
0000     2          .IDENT  'V04-000'
0000     3          .SBTTL  Copyright notice
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26  ;
0000    27
```

```
0000     29          .SBTTL  Program description
0000     30 ;++
0000     31 ;
0000     32 ; FACILITY:
0000     33 ;
0000     34 ;       VAX/VMS SHOW CLUSTER utility
0000     35 ;
0000     36 ;
0000     37 ; ABSTRACT:
0000     38 ;
0000     39 ;       SHWCLSTR is a utility used to display infomation about CI clusters and
0000     40 ;       the local hardware interface to the cluster.  Contained within are the
0000     41 ;       routines necessary to extract this information from the SCS database
0000     42 ;       and the CI port database and present the information to the user in a
0000     43 ;       readable format.  Furthermore, all routines needed to parse and process
0000     44 ;       input from the user are also included here.
0000     45 ;
0000     46 ;
0000     47 ; AUTHOR:       Paul R. DeStefano,      July, 1983
0000     48 ;
0000     49 ;
0000     50 ; MODIFIED BY:
0000     51 ;
0000     52 ;       V03-016 PRD0111         Paul R. DeStefano       31-Jul-1984
0000     53 ;               Update FIRST_DATA_LINE, which contains the number of
0000     54 ;               first line of the scrolling region, when the CLUSTER
0000     55 ;               class is added or removed from the display because
0000     56 ;               the ADD or REMOVE field command was entered and the
0000     57 ;               field was the first CLUSTER class field being added
0000     58 ;               or the last being removed.
0000     59 ;
0000     60 ;       V03-015 PRD0107         Paul R. DeStefano       20-Jul-1984
0000     61 ;               Correct formating of descriptor passed to $GETDVI.
0000     62 ;               Make INIT assume that output goes to a listing file
0000     63 ;               if $GETDVI fails.
0000     64 ;               Make sure initialization file gets executed when
0000     65 ;               running in a Batch job.
0000     66 ;               Change scrolling back to original design.
0000     67 ;
0000     68 ;       V03-014 PRD0103         Paul R. DeStefano       11-Jul-1984
0000     69 ;               Make sure DISPLAY_COMMON assumes successful return.
0000     70 ;               Fix access violation in OUTPUT_TIME   utine.
0000     71 ;
0000     72 ;       V03-013 PRD0100         Paul R. DeStefano       08-May-1984
0000     73 ;               Modified RESET routine to properly restore the init
0000     74 ;               flag.
0000     75 ;
0000     76 ;       V03-012 PRD0098         Paul R. DeStefano       30-Apr-1984
0000     77 ;               Fix DO_SNAPSHOT routine so that variable length buffers
0000     78 ;               are unlocked.
0000     79 ;
0000     80 ;       V03-011 PRD0095         Paul R. DeStefano       11-Apr-1984
0000     81 ;               Removed COORD_STATUS and LOCAL_STATUS fields.
0000     82 ;
0000     83 ;       V03-010 PRD0089         Paul R. DeStefano       29-Mar-1984
0000     84 ;               Fix display of CLUSTER class title when the ADD CLUSTER
0000     85 ;               command is used in an initialization file.
```

```
0000     86 ;
0000     87 ;       V03-009 PRD0080         Paul R. DeStefano       25-Mar-1984
0000     88 ;               Store only the prefix of offset name to save space in
0000     89 ;               field descriptors.
0000     90 ;               Added code to format a 15 character time string to
0000     91 ;               FORMAT_DATA routines.
0000     92 ;               Improved error messages.
0000     93 ;               Improve output of time.
0000     94 ;               Improve method of limiting up- and left-scrolling.
0000     95 ;               Add SECOND_CHANCE routine.
0000     96 ;               Limit refreshing on commands that fail.
0000     97 ;               Added TRANS_TIME and QD_NAME fields.
0000     98 ;               Removed FORM_STATUS field.
0000     99 ;
0000    100 ;       V03-008 PRD0079         Paul R. DeStefano       11-Mar-1984
0000    101 ;               Don't do refresh after SET INTERVAL or SET SCREEN commands.
0000    102 ;               Correct output when using SYS$OUTPUT is redefined using
0000    103 ;               DEFINE/USER.  Correct output when I/O is to a mailbox.
0000    104 ;               Correct output when /OUTPUT is used within a BATCH job.
0000    105 ;               Correct maximum data width when ASCIC format is used with
0000    106 ;               a display routine other than DISPLAY_FIELD.
0000    107 ;
0000    108 ;       V03-007 PRD0076         Paul R. DeStefano       28-Feb-1984
0000    109 ;               Correct output file name when SYS$OUTPUT is defined as a
0000    110 ;               file.
0000    111 ;               Add new fields to CLUSTER and MEMBERS classes.
0000    112 ;               Make sure Local System Block is displayed when VMSVAXCLUSTER
0000    113 ;               doesn't equal 1.
0000    114 ;               Make sure line drawing and special graphics character are set
0000    115 ;               in G1 area in DECcrt on each screen update.
0000    116 ;               Call SET_REGION_FULL before exiting for foreign terminals
0000    117 ;               emulating VT100s.
0000    118 ;               Perform initial syntax checking using DCL_PARSE.
0000    119 ;               Improved parsing of lists of parameters and qualifier values.
0000    120 ;               Added ADD_ALL_FIELDS routine for new ADD class/ALL qualifier.
0000    121 ;               Make INTERVAL value = seconds rather than delta time.
0000    122 ;               Improved message handling.
0000    123 ;               Modified REMOVE_FIELD routine to turn off a class when the
0000    124 ;               last field is removed.
0000    125 ;               Display CLUSTER class at the top of the CLUSTER report.
0000    126 ;
0000    127 ;       V03-006 PRD0046         Paul R. Destefano       11-Jan-1984
0000    128 ;               Correct the number of bytes displayed by the SYS_ID field.
0000    129 ;               Make COUNTERS a separate class.
0000    130 ;               Retry snapshot with expanded buffers if snapshot buffers
0000    131 ;               overflow.
0000    132 ;
0000    133 ;       V03-005 PRD0043         Paul R. DeStefano       27-Sep-1983
0000    134 ;               Initialize connection count in path blocks at
0000    135 ;               PB$SHC$L_NUM_CDT so that a circuit losing all
0000    136 ;               connections will be properly detected.
0000    137 ;               Fix access violation in CHECK_POS routine.
0000    138 ;
0000    139 ;       V03-004 PRD0042         Paul R. DeStefano       27-Sep-1983
0000    140 ;               Fix setup of initialization file default so that an
0000    141 ;               extention of .INI is defaulted but not forced.
0000    142 ;               Add additional remote port types to RP_TYPE field.
```

```
0000   143 ;
0000   144 ;       V03-003 PRD0041         Paul R. DeStefano       15-Sep-1983
0000   145 ;               Modified default report display.
0000   146 ;               Fix handling of formative circuits in CHECK_POS routine.
0000   147 ;               Add variable buffer address (VAR_BUF_ADRS) to locked
0000   148 ;               portion of working set.
0000   149 ;               Fix refresh after INITIALIZE command.
0000   150 ;
0000   151 ;       V03-002 PRD0028         Paul R. DeStefano       23-Aug-1983
0000   152 ;               Modified handling of broadcast messages to allow for
0000   153 ;               messages that lack carriage control characteristics.
0000   154 ;
0000   155 ;               Fixed handling of an initialization command file that
0000   156 ;               contains an INITIALIZE command.
0000   157 ;
0000   158 ;--
0000   159
```

```
0000    161
0000    162          .SBTTL  Definitions.
0000    163 ;
0000    164 ; System definitions: (From SYS$LIBRARY:LIB.MLB)
0000    165 ;
0000    166
0000    167          $CDLDEF                         ; Define connection descriptor list
0000    168                                          ;  offsets.
0000    169          $CDTDEF                         ; Define connection descriptor table
0000    170                                          ;  offsets.
0000    171          $CIBDDEF                        ; Define buffer descriptor offsets.
0000    172          $CIBDTDEF                       ; Define buffer descriptor table
0000    173                                          ;  offsets.
0000    174          $CLUBDEF                        ; Define Cluster Block.
0000    175          $CSBDEF                         ; Define Cluster System Block.
0000    176          $DCDEF                          ; Define device class codes.
0000    177          $DDBDEF                         ; Define DDB offsets.
0000    178          $DIBDEF                         ; Define device characteristics offsets.
0000    179          $DVIDEF                         ; Define item identifier codes.
0000    180          $HLPDEF                         ; Define help flags.
0000    181          $IPLDEF                         ; Define processor priority levels.
0000    182          $MSGDEF                         ; Define message types.
0000    183          $PBDEF                          ; Define path block offsets.
0000    184          $PDTDEF                         ; Define port descriptor table offsets.
0000    185          $PRVDEF                         ; Define privilege mask bits.
0000    186          $SBDEF                          ; Define system block offsets.
0000    187          $SCRDEF                         ; Define screen package parameters.
0000    188          $SCSDEF                         ; Define SCS message formats.
0000    189          $SDIRDEF                        ; Define SCS directory entry offsets.
0000    190          $SSDEF                          ; Define status codes.
0000    191          $TPADEF                         ; Define LIB$TPARSE parameter block.
0000    192          $TRMDEF                         ; Define terminal function modifiers.
0000    193          $TTDEF                          ; Define terminal characteristics.
0000    194          $TT2DEF                         ; Define extended terminal
0000    195                                          ;  characteristics.
0000    196          $UCBDEF                         ; Define UCB offsets.
0000    197
0000    198 ;
0000    199 ; CI specific definitions: (From PALIB.MLB)
0000    200 ;
0000    201
0000    202          $PAPDTDEF                       ; Define the extention to the port
0000    203                                          ;  descriptor table.
0000    204          $PPDDEF                         ; Define datagram offsets.
0000    205
0000    206 ;
0000    207 ; Define offsets into field descriptors.
0000    208 ;
0000    209
0000    210          $DEFINI FD
0000    211
0000    212          $DEF    FD$B_FIELD_SIZE         .BYTE   ; Number of columns currently
0001    213                                                  ;  occupied by the field.
0001    214          $DEF    FD$B_MIN_FLD_SIZE       .BYTE   ; Minimum number of columns that
0002    215                                                  ;  can be used to display the
0002    216                                                  ;  field.
0002    217          $DEF    FD$B_MAX_DATA_SIZE      .BYTE   ; Maximum number of columns
```

```
                     0003   218                                                    ;  needed to display the data.
                     0003   219          $DEF    FD$B_MAX_FLD_SIZE       .BYTE   ; Maximum number of columns
                     0004   220                                                    ;  needed to display the field.
                     0004   221                                                    ;  (For both title and data.)
                     0004   222          $DEF    FD$B_LENGTH             .BYTE   ; Number of bytes of data to
                     0005   223                                                    ;  extract from the SCS
                     0005   224                                                    ;  database to display the
                     0005   225                                                    ;  field.
                     0005   226          $DEF    FD$B_FORMAT             .BYTE   ; Type of format to use when
                     0006   227                                                    ;  displaying the field.
                     0006   228                                                    ;
            00000008 0006   229 .=.+2                                            ; Reserve one word.
                     0008   230                                                    ;
                     0008   231          $DEF    FD$L_DISPLAY_ROUTINE    .LONG   ; Routine to call to display the
                     000C   232                                                    ;  field.
                     000C   233          $DEF    FD$L_CLASS              .LONG   ; Class to which the field
                     0010   234                                                    ;  belongs.
                     0010   235          $DEF    FD$L_OFFSET             .LONG   ; Value of the SCS offset.
                     0014   236          $DEF    FD$L_OFFSET_PRFX        .LONG   ; First four characters of
                     0018   237                                                    ;  offset name string.
                     0018   238          $DEF    FD$B_TITLE_SIZE         .BYTE   ; Number of characters in the
                     0019   239                                                    ;  title for the field.
                     0019   240          $DEF    FD$T_TITLE                      ; Offset to the title string.
                     0019   241
                     0019   242          $DEFEND FD,,DEF
                     0000   243
                     0000   244
                     0000   245 ;
                     0000   246 ; Define the field display format types.
                     0000   247 ;
                     0000   248
            00000001 0000   249 ASCII           =       1               ; Data is to be formatted in ascii.
            00000002 0000   250 HEX             =       2               ; Data is to be formatted in hexadecimal
            00000003 0000   251 DEC             =       3               ; Data is to be formatted in decimal.
            00000004 0000   252 ASCIC           =       4               ; Data is to be formatted using a
                     0000   253                                         ;  counted ascii string.
            00000005 0000   254 TIME            =       5               ; Data is to be formatted as a time
                     0000   255                                         ;  string.
            0000000F 0000   256 TIME_STRING_LEN =       15              ; Length of formatted time string.
                     0000   257
                     0000   258 ;
                     0000   259 ; Define lengths and starting offsets of special buffers for snapshot data.
                     0000   260 ;
                     0000   261
                     0000   262 ;
                     0000   263 ; The portion of the CDT stored starts at CDT$L_RCONID.  The length is the
                     0000   264 ; number of bytes in CDT$L_RCONID through CDT$W_QBDT_CNT plus the forward link.
                     0000   265 ; The difference is a negative value to be added to the SCS offset in the
                     0000   266 ; field descriptors to compensate for the portion of the CDT that isn't stored.
                     0000   267 ;
                     0000   268
            00000014 0000   269 SHC$L_CDT_BEGIN =       CDT$L_RCONID                    ; Starting offset.
            0000008C 0000   270 SHC$C_CDT_LENGTH=       <<CDT$W_QBDT_CNT+2>-CDT$L_RCONID>+4      ; Length.
            FFFFFFF0 0000   271 SHC$L_CDT_DIF   =       <CDT$L_MSGINPUT-CDT$L_RCONID>+4 ; Difference.
                     0000   272
                     0000   273 ;
                     0000   274 ; The portion of the PDT stored starts at PDT$L_MSGHDRSZ.  The length is the
```

```
                     0000   275 ; number of bytes in PDT$L_MSGHDRSZ through PDT$L_LBDG.  The difference is a
                     0000   276 ; negative value to be added to the SCS offset in the field descriptors to
                     0000   277 ; compensate for the portion of the PDT that isn't stored.
                     0000   278
00000064             0000   279 SHC$L_PDT_BEGIN =       PDT$L_MSGHDRSZ                  ; Starting offset.
000000D0             0000   280 SHC$L_PDT_LENGTH=       PDT$L_LBDG-PDT$L_MSGHDRSZ       ; Length.
FFFFFF4C             0000   281 SHC$L_PDT_DIF   =       <PDT$Q_SIZE-8>-PDT$L_MSGHDRSZ   ; Difference.
                     0000   282
                     0000   283
                     0000   284 ;
                     0000   285 ; Define offsets for storing the number of circuits on a system and the number
                     0000   286 ; of connections on a circuit.
                     0000   287 ;
                     0000   288
00000004             0000   289 SB$SHC$L_NUM_PB =       SB$L_BLINK      ; Store the number of paths block on a
                     0000   290                                         ;  system block where the back link
                     0000   291                                         ;  would normally be.
00000004             0000   292 PB$SHC$L_NUM_CDT=       PB$L_BLINK      ; Store the number of connections on a
                     0000   293                                         ;  path block where the back link would
                     0000   294                                         ;  normally be.
                     0000   295 ;
                     0000   296 ; Define the offset for storing the display's line number in the system block.
                     0000   297 ; This is used to associate a system with a line in the display to insure that
                     0000   298 ; a line which has moved in the display, but has not changed does not go
                     0000   299 ; undetected.
                     0000   300 ;
                     0000   301
00000009             0000   302 SHC$SB'  _DIS_POS=       SB$W_SIZE+1     ; Store in upper byte of size field in
                     0000   303                                         ;  system block.
                     0000   304
                     0000   305 ;
                     0000   306 ; Define offsets into MISC_BUF.
                     0000   307 ;
                     0000   308
                     0000   309         $DEFINI MB
                     0000   310
                     0000   311         $DEF    MB$L_PORT_NAME   .LONG  ; Port name.
                     0004   312         $DEF    MB$L_PORT_NUM    .LONG  ; Port number.
                     0008   313         $DEF    MB$L_DEVTYPE     .LONG  ; Device type.
                     000C   314         $DEF    MB$L_STS         .LONG  ; Device status.
                     0010   315         $DEF    MB$B_ERTCNT      .BYTE  ; # of reinitialize attempts remaining.
                     0011   316         $DEF    MB$B_ERTMAX      .BYTE  ; Maximum number of reinits allowed.
                     0012   317         $DEF    MB$W_ERRCNT      .WORD  ; Number of errors logged.
                     0014   318         $DEF    MB$L_FORM_PB     .LONG  ; Number of formative path blocks.
                     0018   319         $DEF    MB$L_PL_WAITERS  .LONG  ; Number of SCS non-paged pool waiters.
                     001C   320         $DEF    MB$W_DG_FREE     .WORD  ; Number of free datagram buffers
                     001E   321                                         ;  currently queued for receives.
                     001E   322         $DEF    MB$W_MSG_FREE    .WORD  ; Number of free message buffers
                     0020   323                                         ;  currently queued for receives.
                     0020   324         $DEF    MB$W_CMDS_QUED   .WORD  ; Total number of messages, datagrams,
                     0022   325                                         ;  and port commands queued for
                     0022   326                                         ;  transmission at all priorities.
                     0022   327                                         ;  by this port.
                     0022   328         $DEF    MB$W_RSPS_QUED   .WORD  ; Number of responses of all kinds
                     0024   329                                         ;  received on this port, but not yet
                     0024   330                                         ;  processed.
                     0024   331         $DEFEND MB,,DEF
```

```
                     0000   332
0000002E             0000   333 MISC_BUF_LEN    =       46              ; 11 longwords in miscellaneous buffer.
                     0000   334
                     0000   335
                     0000   336 ;
                     0000   337 ; Define masks and bit fields for device types.
                     0000   338 ;
                     0000   339
00000000             0000   340 DVT$V_VT52       =       0               ; VT52 compatable terminal.
00000001             0000   341 DVT$V_DCRT       =       1               ; VT100 compatable terminal.
00000002             0000   342 DVT$V_HARD       =       2               ; Hardcopy device.
00000003             0000   343 DVT$V_UNKNOWN    =       3               ; Unknown type of scope.
00000004             0000   344 DVT$V_AVO        =       4               ; Advanced video option present.
00000005             0000   345 DVT$V_DISK       =       5               ; Disk.
                     0000   346
00000001             0000   347 DVT$M_VT52       =       1@DVT$V_VT52
00000002             0000   348 DVT$M_DCRT       =       1@DVT$V_DCRT
00000004             0000   349 DVT$M_HARD       =       1@DVT$V_HARD
00000008             0000   350 DVT$M_UNKNOWN    =       1@DVT$V_UNKNOWN
00000010             0000   351 DVT$M_AVO        =       1@DVT$V_AVO
00000020             0000   352 DVT$M_DISK       =       1@DVT$V_DISK
                     0000   353
                     0000   354 ;
                     0000   355 ; Define the terminal screen widths.
                     0000   356 ;
                     0000   357
00000050             0000   358 NARROW           =       80              ; DECcrt 80 column mode.
00000084             0000   359 WIDE             =       132             ; DECcrt 132 column mode.
                     0000   360
                     0000   361
                     0000   362 ;
                     0000   363 ; Define special keyboard characters.
                     0000   364 ;
                     0000   365
0000001B             0000   366 ESC              =       ^X1B            ; Ascii escape character.
0000009B             0000   367 CSI              =       ^X9B            ; Eight bit equivalent of "ESC [".
00000017             0000   368 CTRLW            =       ^X17            ; Ascii control-W.
0000001A             0000   369 CTRLZ            =       ^X1A            ; Ascii control-Z.
                     0000   370
                     0000   371
                     0000   372 ;
                     0000   373 ; Define the escape sequences sent by the arrow keys.
                     0000   374 ;
                     0000   375
0000415B             0000   376 DCRT_UP_ARROW    =       ^A/[A/          ; DECcrt up arrow.
0000425B             0000   377 DCRT_DWN_ARROW   =       ^A/[B/          ; DECcrt down arrow.
0000435B             0000   378 DCRT_LFT_ARROW   =       ^A/[C/          ; DECcrt left arrow.
0000445B             0000   379 DCRT_RT_ARROW    =       ^A/[D/          ; DECcrt right arrow.
                     0000   380
00000041             0000   381 VT52_UP_ARROW    =       ^A/A/           ; VT52 up arrow.
00000042             0000   382 VT52_DWN_ARROW   =       ^A/B/           ; VT52 down arrow.
00000043             0000   383 VT52_LFT_ARROW   =       ^A/C/           ; VT52 left arrow.
00000044             0000   384 VT52_RT_ARROW    =       ^A/D/           ; VT52 right arrow.
                     0000   385
                     0000   386
                     0000   387 ;
                     0000   388 ; Special characters to turn the special graphics characters on and off.
```

```
                     0000   389 ;
                     0000   390
0000000E             0000   391 GRAPHICS_ON     =       ^X0E            ; Control_N or SO (shift-out)
                     0000   392
0000000F             0000   393 GRAPHICS_OFF    =       ^X0F            ; Control_O or SI (shift-in)
                     0000   394
                     0000   395
                     0000   396 ;
                     0000   397 ; DECcrt special graphics characters
                     0000   398 ;
                     0000   399
0000006A             0000   400 LR_CORNER       =       ^A/j/           ; Lower-right corner.
0000006B             0000   401 UR_CORNER       =       ^A/k/           ; Upper-right corner.
0000006C             0000   402 UL_CORNER       =       ^A/l/           ; Upper-left corner.
0000006D             0000   403 LL_CORNER       =       ^A/m/           ; Lower-left corner.
0000006E             0000   404 CROSS           =       ^A/n/           ; Crossing lines.
00000071             0000   405 SCAN_5          =       ^A/q/           ; Horizontal line - Scan 5.
00000074             0000   406 LEFT_T          =       ^A/t/           ; Left "T".
00000075             0000   407 RIGHT_T         =       ^A/u/           ; Right "T".
00000076             0000   408 BOTTOM_T        =       ^A/v/           ; Bottom "T".
00000077             0000   409 TOP_T           =       ^A/w/           ; TOP "T".
00000078             0000   410 VERT_BAR        =       ^A/x/           ; Vertical bar.
                     0000   411
                     0000   412
                     0000   413 ;
                     0000   414 ; Define masks and bit fields for the type of information currently output
                     0000   415 ; on the two command input lines.
                     0000   416 ;
                     0000   417
00000000             0000   418 CL$V_BRDCST     =       0               ; Broadcast message.
00000001             0000   419 CL$V_CMD        =       1               ; Command input.
00000002             0000   420 CL$V_ERROR      =       2               ; Error message.
00000003             0000   421 CL$V_SCROLL     =       3               ; Scroll message.
                     0000   422
00000001             0000   423 CL$M_BRDCST     =       1@CL$V_BRDCST
00000002             0000   424 CL$M_CMD        =       1@CL$V_CMD
00000004             0000   425 CL$M_ERROR      =       1@CL$V_ERROR
00000008             0000   426 CL$M_SCROLL     =       1@CL$V_SCROLL
                     0000   427
                     0000   428
                     0000   429
                     0000   430 ;
                     0000   431 ; Define commonly used line numbers.
                     0000   432 ;
                     0000   433
00000002             0000   434 HEADER_START    =       2               ; Display header starts at line two.
00000007             0000   435 HEADER_FND      =       HEADER_START+5  ;  and is five lines long.
                     0000   436
                     0000   437
                     0000   438 ;
                     0000   439 ; Define the mask for CURRENT_REPORT.
                     0000   440 ;
                     0000   441
00000000             0000   442 RPT$V_CLSTR     =       0               ; Set for cluster report, clear for
00000001             0000   443 RPT$M_CLSTR     =       1@RPT$V_CLSTR   ;  local report.
                     0000   444
                     0000   445
```

```
                     0000   446 ;
                     0000   447 ; Define ASCIC strings used to identify device descriptor blocks.
                     0000   448 ;
                     0000   449
41415003             0000   450 PAAC            =       ^X41415003      ; "PAA".
42415003             0000   451 PABC            =       ^X42415003      ; "PAB"
                     0000   452
                     0000   453
                     0000   454 ;
                     0000   455 ; To add a new class.
                     0000   456 ;
                     0000   457 ;       1)      Define the class mask and field bits.
                     0000   458 ;       2)      Add the ascii class name @CLASS_NAMES
                     0000   459 ;       3)      Add a longword to the TITLE_LEN_ARRAY
                     0000   460 ;       4)      Add field definitions preceeding them with a
                     0000   461 ;               CLASS = class_mask statement.
                     0000   462 ;       5)      Increase NUM_CLASSES by 1.
                     0000   463 ;
                     0000   464
                     0000   465 ;
                     0000   466 ; To add a new data structure for display by DISPLAY_FIELD.
                     0000   467 ;
                     0000   468 ;       1)      Add the new prefix to SYM_PRFX.
                     0000   469 ;       2)      Add a buffer for the new structure (fixed length) or
                     0000   470 ;               descriptors for a variable buffer.
                     0000   471 ;       3)      Modify the SNAPSHOT routine to capture the new structure.
                     0000   472 ;       4)      Add a new GET_DATA call frame index.
                     0000   473 ;
                     0000   474
00000009             0000   475 NUM_CLASSES     =       9               ; Nine basic class types.
                     0000   476
                     0000   477 ;
                     0000   478 ; Masks for controlling display of classes.
                     0000   479 ;
                     0000   480
00000000             0000   481 CLS$V_CLUB      =       0               ; Cluster class.
00000001             0000   482 CLS$V_SYS       =       1               ; Systems class.
00000002             0000   483 CLS$V_CSB       =       2               ; Members class.
00000003             0000   484 CLS$V_CIR       =       3               ; Circuits class.
00000004             0000   485 CLS$V_CON       =       4               ; Connections class.
00000005             0000   486 CLS$V_CNT       =       5               ; Counters class.
00000006             0000   487 CLS$V_CRD       =       6               ; Credits class.
00000007             0000   488 CLS$V_LPRT      =       7               ; Local ports class.
00000008             0000   489 CLS$V_ERR       =       8               ; Errors class.
00000009             0000   490 CLS$V_OPEN_CIR  =       9               ; Circuits in the open state.
0000000A             0000   491 CLS$V_NOOPEN_CIR=       10              ; Circuits in states other than open.
0000000B             0000   492 CLS$V_OPEN_CON  =       11              ; Connections in the open state.
0000000C             0000   493 CLS$V_NOOPEN_CON=       12              ; Connections in states other than open.
                     0000   494
                     0000   495 ;
                     0000   496 ; Types of systems.  Must match HW_TYPE_TABLE.
                     0000   497 ;
                     0000   498
00000010             0000   499 CLS$V_750_SYS   =       16              ; 750 systems.
00000011             0000   500 CLS$V_780_SYS   =       17              ; 780 systems.
00000012             0000   501 CLS$V_785_SYS   =       18              ; 785 systems.
00000013             0000   502 CLS$V_790_SYS   =       19              ; 790 systems.
```

```
                                                B  4

00000014  0000    503 CLS$V_HSC50_SYS =       20              ; HSC50 systems.
00000015  0000    504 CLS$V_HSC60_SYS =       21              ; HSC60 systems.
          0000    505
00000001  0000    506 CLS$M_CLUB      =       1@CLS$V_CLUB
00000002  0000    507 CLS$M_SYS       =       1@CLS$V_SYS
00000004  0000    508 CLS$M_CSB       =       1@CLS$V_CSB
00000008  0000    509 CLS$M_CIR       =       1@CLS$V_CIR
00000010  0000    510 CLS$M_CON       =       1@CLS$V_CON
00000020  0000    511 CLS$M_CNT       =       1@CLS$V_CNT
00000040  0000    512 CLS$M_CRD       =       1@CLS$V_CRD
00000080  0000    513 CLS$M_LPRT      =       1@CLS$V_LPRT
00000100  0000    514 CLS$M_ERR       =       1@CLS$V_ERR
          0000    515
          0000    516 ;
          0000    517 ; Mask for any type of SCS class. (Includes MEMBERS)
          0000    518 ;
          0000    519
          0000    520 CLS$M_ANY_SCS   -       <CLS$M_SYS!-    ; SYSTEMS
          0000    521                          CLS$M_CIR!-    ; CIRCUITS
          0000    522                          CLS$M_CON!-    ; CONNECTIONS
          0000    523                          CLS$M_CNT!-    ; COUNTERS
          0000    524                          CLS$M_CRD!-    ; CREDITS
0000007E  0000    525                          CLS$M_CSB>     ; MEMBERS
          0000    526
          0000    527 ;
          0000    528 ; Mask for any type of CLUSTER report class.
          0000    529 ;
          0000    530
          0000    531 CLS$M_ANY_CLUS  =       <CLS$M_ANY_SCS!-        ; Any SCS class.
0000007F  0000    532                          CLS$M_CLUB>            ; Plus the CLUSTER class.
          0000    533
          0000    534 ;
          0000    535 ; Mask for any types of system
          0000    536 ;
          0000    537
00020000  0000    538 CLS$M_780_SYS   =       1@CLS$V_780_SYS         ; 11/780
00040000  0000    539 CLS$M_785_SYS   =       1@CLS$V_785_SYS         ; 11/785
00010000  0000    540 CLS$M_750_SYS   =       1@CLS$V_750_SYS         ; 11/750
00100000  0000    541 CLS$M_HSC50_SYS =       1@CLS$V_HSC50_SYS       ; HSC50
00200000  0000    542 CLS$M_HSC60_SYS =       1@CLS$V_HSC60_SYS       ; HSC60
          0000    543
          0000    544
          0000    545 CLS$M_ALL_SYS   =       <CLS$M_SYS!-            ; Any type of system...
          0000    546                          CLS$M_780_SYS!-        ; ...11/780
          0000    547                          CLS$M_785_SYS!-        ; ...11/785
          0000    548                          CLS$M_750_SYS!-        ; ...11/750
          0000    549                          CLS$M_HSC50_SYS!-      ; ...HSC50
00370002  0000    550                          CLS$M_HSC60_SYS>       ; ...HSC60
          0000    551
          0000    552
          0000    553 ;
          0000    554 ; Mask for all types of circuits.
          0000    555 ;
          0000    556
00000200  0000    557 CLS$M_OPEN_CIR  =       1@CLS$V_OPEN_CIR        ; Open CIRCUITS.
00000400  0000    558 CLS$M_NOOPEN_CIR=       1@CLS$V_NOOPEN_CIR      ; Non-open CIRCUITS.
          0000    559
```

```
                     0000   560 CLS$M_ALL_CIR   =       <CLS$M_CIR!-              ; All types of circuits...
                     0000   561                          CLS$M_OPEN_CIR!-          ; ...Open CIRCUITS
00000608             0000   562                          CLS$M_NOOPEN_CIR>         ; ...Non-open CIRCUITS
                     0000   563
                     0000   564 ;
                     0000   565 ; Masks for all types of connections.
                     0000   566 ;
                     0000   567
00000800             0000   568 CLS$M_OPEN_CON  =       1@CLS$V_OPEN_CON          ; Open CONNECTIONS
00001000             0000   569 CLS$M_NOOPEN_CON=       1@CLS$V_NOOPEN_CON        ; Non-open CONNECTIONS
                     0000   570
                     0000   571 CLS$M_ALL_CON   =       <CLS$M_CON!-              ; All types of CONNECTIONS...
                     0000   572                          CLS$M_OPEN_CON!-          ; ...Open CONNECTIONS
00001810             0000   573                          CLS$M_NOOPEN_CON>         ; ...Non-open CONNECTIONS
                     0000   574
                     0000   575 ;
                     0000   576 ; Define the longword indices into the GET_DATA call frame.
                     0000   577 ;
                     0000   578
00000001             0000   579 CLUB_IN          =       1
00000002             0000   580 SB_IN            =       2
00000003             0000   581 CSB_IN           =       3
00000004             0000   582 PB_IN            =       4
00000005             0000   583 CDT_IN           =       5
00000006             0000   584 PDT_IN           =       6
00000007             0000   585 MB_IN           =       7
                     0000   586
                     0000   587 ;
                     0000   588 ; Masks for event flags.
                     0000   589 ;
                     0000   590
00000001             0000   591 EVF$V_TIM_FLAG   =       1                         ; Refresh timer expired.
00000002             0000   592 EVF$V_AST_FLAG   =       2                         ; AST delivery.
                     0000   593
00000002             0000   594 EVF$M_TIM_FLAG   =       1@EVF$V_TIM_FLAG
00000004             0000   595 EVF$M_AST_FLAG   =       1@EVF$V_AST_FLAG
                     0000   596
                     0000   597 ;
                     0000   598 ; Miscellaneous flags.
                     0000   599 ;
                     0000   600
00000000             0000   601 MF$V_INIT_FLAG   =       0                         ; Initialization in progress.
00000001             0000   602 MF$V_EXIT_FLAG   =       1                         ; User requested exit.
00000002             0000   603 MF$V_INVERSE     =       2                         ; Field is to be displayed in inverse
                     0000   604                                                   ;  video.
00000003             0000   605 MF$V_RIP         =       3                         ; Indicates that a refresh is in
                     0000   606                                                   ;  progress.
00000004             0000   607 MF$V_CONT        =       4                         ; Indicates that the display is to be
                     0000   608                                                   ;  updated continuously.
00000005             0000   609 MF$V_CYCLE       =       5                         ; Indicates that the display is to be
                     0000   610                                                   ;  cycled every other update.
00000006             0000   611 MF$V_IN_OPEN     =       6                         ; Indicates that a command procedure
                     0000   612                                                   ;  file is open.
00000007             0000   613 MF$V_NEW_CHAR    =       7                         ; Indicates terminal characteristics
                     0000   614                                                   ;  have been modified.
00000008             0000   615 MF$V_SCROLL      =       8                         ; Indicates that the screen has been
                     0000   616                                                   ;  scrolled.
```

```
00000009  0000   617 MF$V_VALID_MSG  =       9               ; Indicates valid ascii string found in
          0000   618                                         ;  message buffer.
0000000A  0000   619 MF$V_CR_FOUND   =       10              ; Indicates that a valid ascii string
          0000   620                                         ;  followed by a carriage return has
          0000   621                                         ;  been found in the message buffer.
          0000   622
00000001  0000   623 MF$M_INIT_FLAG  =       1@MF$V_INIT_FLAG
00000002  0000   624 MF$M_EXIT_FLAG  =       1@MF$V_EXIT_FLAG
00000004  0000   625 MF$M_INVERSE    =       1@MF$V_INVERSE
00000008  0000   626 MF$M_RIP        =       1@MF$V_RIP
00000010  0000   627 MF$M_CONT       =       1@MF$V_CONT
00000020  0000   628 MF$M_CYCLE      =       1@MF$V_CYCLE
00000040  0000   629 MF$M_IN_OPEN    =       1@MF$V_IN_OPEN
00000080  0000   630 MF$M_NEW_CHAR   =       1@MF$V_NEW_CHAR
00000100  0000   631 MF$M_SCROLL     =       1@MF$V_SCROLL
00000200  0000   632 MF$M_VALID_MSG  =       1@MF$V_VALID_MSG
00000400  0000   633 MF$M_CR_FOUND   =       1@MF$V_CR_FOUND
          0000   634
          0000   635 ;
          0000   636 ; Scroll flags.
          0000   637 ;
          0000   638
00000000  0000   639 SCR$V_MORE_RT   =       0               ; Data has truncated off right of
          0000   640                                         ;  screen.
00000001  0000   641 SCR$V_MORE_LFT  =       1               ; Display has been scrolled left.
00000002  0000   642 SCR$V_MORE_UP   =       2               ; Display has been scrolled up.
00000003  0000   643 SCR$V_MORE_DWN  =       3               ; Data has been truncated off the
          0000   644                                         ;  bottom of the screen.
00000004  0000   645 SCR$V_MORE      =       4               ; Any "more" flag has been set.
          0000   646
00000001  0000   647 SCR$M_MORE_RT   =       1@SCR$V_MORE_RT
00000002  0000   648 SCR$M_MORE_LFT  =       1@SCR$V_MORE_LFT
00000004  0000   649 SCR$M_MORE_UP   =       1@SCR$V_MORE_UP
00000008  0000   650 SCR$M_MORE_DWN  =       1@SCR$V_MORE_DWN
00000010  0000   651 SCR$M_MORE      =       1@SCR$V_MORE
          0000   652
          0000   653
00000010  0000   654 MAX_ID          =       16              ; Maximum number of system that can be
          0000   655                                         ;  removed from the display using
          0000   656                                         ;  REMOVE SYSTEM/ID=system_id.
00000002  0000   657 MAXPDT          =       2               ; Maximum number of port descriptor
          0000   658                                         ; tables per system.
00000003  0000   659 QUE_LIMIT       =       3               ; Maximum number of entries to check
          0000   660                                         ;  in an interlocked queue.
00000012  0000   661 DEF_CONNCNT     =       18              ; Default value for the SYSGEN
          0000   662                                         ;  parameter SCSCONNCNT.
          0000   663
          0000   664         .SBTTL  Ascii strings.
```

```
                        00000000   666          .PSECT  RODATA,RD,NOWRT,NOEXE
                              0000 667
                              0000 668 TIME_SOURCE:
                        00    0000 669          .BYTE   0
                              0001 670
                              0001 671 ;
                              0001 672 ; Filespecs and logical names for initialization.
                              0001 673 ;
                              0001 674
                              0001 675 DEF_EXT:                                ; Default file extention for
                     49 4E 49 2E  0001 676          .ASCII  /.INI/             ;  initialization if SHOW_CLUSTER$INIT
                              0005 677                                         ;  is defined.
                        00000004  0005 678 DEF_EXT_LEN     =.-DEF_EXT         ; Length of string.
                              0005 679
                              0005 680 DEF_NAME2:                              ; Default command procedures filename
52 45 54 53 55 4C 43 5F 57 4F 48 53  0005 681          .ASCII  /SHOW_CLUSTER$INIT:.COM/;  if SHOW_CLUSTER$INIT is defined.
      4D 4F 43 2E 3A 54 49 4E 49 24  0011
                        00000016  001B 682 DEF_NAME2_LEN   =.-DEF_NAME2       ; Length of string.
                              001B 683
                              001B 684 ;
                              001B 685 ; Names of qualifiers to the DCL level SHOW command.
                              001B 686 ;
                              001B 687
                              001B 688 REPORT_QUAL:                            ; Qualifier for specifying report name.
54 52 4F 50 45 52 00000023'010E0000'  001B 689          .ASCID  /REPORT/
                              0029 690
                              0029 691 CONTINUOUS_QUAL:                        ; Qualifier for specifying continuous
4E 49 54 4E 4F 43 00000031'010E0000'  0029 692          .ASCID  /CONTINUOUS/     ;  update display.
                     53 55 4F 55  0037
                              003B 693
                              003B 694 INTERVAL_QUAL:                          ; Qualifier for specifying interval
56 52 45 54 4E 49 00000043'010E0000'  003B 695          .ASCID  /INTERVAL/       ;  between display updates.
                           4C 41  0049
                              004B 696
                              004B 697 OUTPUT_QUAL:                            ; Qualifier for specifying output file.
54 55 50 54 55 4F 00000053'010E0000'  004B 698          .ASCID  /OUTPUT/         ;
                              0059 699
                              0059 700 ENDING_QUAL:                            ; Qualifier for specifying ending time.
47 4E 49 44 4E 45 00000061'010E0000'  0059 701          .ASCID  /ENDING/         ;
                              0067 702
                              0067 703 BEGIN_QUAL:                             ; Qualifier for specifying starting
4E 4E 49 47 45 42 0000006F'010E0000'  0067 704          .ASCID  /BEGINNING/      ;  time.
                        47 4E 49  0075
                              0078 705
                              0078 706 ;
                              0078 707 ; Ascii strings used to make up the scrolling signals.
                              0078 708 ;
                              0078 709
                              0078 710 MORE_DSCR:
20 20 20 09 09 09 00000080'010E0000'  0078 711          .ASCID  /                    More    /
      20 20 20 20 65 72 6F 4D 20 20  0086
                              0090 712 MORE_TABS:                              ; Extra tabs for 132 column mode.
09 09 09 09 09 09 00000098'010E0000'  0090 713          .ASCID  /                                                    More    /
         20 20 20 20 65 72 6F 4D 09  009E
         20 20 5E 000000AF'010E0000'  00A7 714 MORE_UP:.ASCID  /^  /              ; Scroll down allowed.
                              00B2 715 MORE_DWN:
         20 20 76 000000BA'010E0000'  00B2 716          .ASCID  /v  /            ; Scroll up allowed.
```

```
                                00BD    717 MORE_LFT:
          2D 2D 3C 000000C5'010E0000'   00BD    718             .ASCID  /<--/                   ; Scroll left allowed.
          3E 2D 2D 000000D0'010E0000'   00C8    719 MORE_RT:.ASCID  /-->/                   ; Scroll right allowed.
          20 20 20 000000DB'010E0000'   00D3    720 NO_MORE:.ASCID  /   /
                                00DE    721
                                00DE    722
                                00DE    723 ;
                                00DE    724 ; Logical names.
                                00DE    725 ;
                                00DE    726
                                00DE    727 IN_DEV_NAME:                            ; Input device name.
4E 49 24 53 59 53 000000E6'010E0000'   00DE    728             .ASCID  /SYS$INPUT/         ;
                             54 55 50   00EC
                                00EF    729
                                00EF    730 OUT_DEV_NAME:                           ; Output device name.
55 4F 24 53 59 53 000000F7'010E0000'   00EF    731             .ASCID  /SYS$OUTPUT/        ;
                          54 55 50 54   00FD
                                0101    732
                                0101    733 NULL_DEV_NAME:                          ; Null device name.
          3A 4C 4E 00000109'010E0000'   0101    734             .ASCID  /NL:/               ;
                                010C    735
                                010C    736 LIBRARY_NAME:                           ; Name of help library for interactive
45 48 24 53 59 53 00000114'010E0000'   010C    737             .ASCID  /SYS$HELP:SHWCLHELP.HLB/;  parser.
50 4C 45 48 4C 43 57 48 53 3A 50 4C   011A
                      42 4C 48 2E   0126
                                012A    738 INIT_NAME:                              ; Startup initialization file name.
43 5F 57 4F 48 53 00000132'010E0000'   012A    739             .ASCID  /SHOW_CLUSTER$INIT/     ;
   54 49 4E 49 24 52 45 54 53 55 4C   0138
                                0143    740
                                0143    741 ;
                                0143    742 ; Terminal control strings.
                                0143    743 ;
                                0143    744
                                0143    745 DCRT_NOATR_DSCR:                        ; DECcrt control string to turn
      6D 5B 1B 0000014B'010E0000'   0143    746             .ASCID  <ESC>/[m/           ;  off character attributes.
                                014E    747
                                014E    748 DCRT_NARROW_DSCR:                       ; Escape sequence used to place DECcrt
6C 33 3F 5B 1B 00000156'010E0000'   014E    749             .ASCID  /[?3l/              ;  in 80 column mode.
                                015B    750 DCRT_WIDE_DSCR:                         ; Escape sequence used to place DECcrt
68 33 3F 5B 1B 00000163'010E0000'   015B    751             .ASCID  /[?3h/              ;  in 132 column mode.
                                0168    752
                                0168    753 GRAPHICS_DSGNTR:                        ; Escape sequence designate special
      30 29 1B 00000170'010E0000'   0168    754             .ASCID  /)0/                ;  graphics and line drawing character
                                0173    755                                         ;  set into G1 area for DECcrt.
                                0173    756
                                0173    757 DELM_DSCR:                              ; Control string to output a
      0F 78 0E 00000178'010E0000'   0173    758             .ASCID  /x/                 ; vertical bar on a DECcrt.
                          00000003   017E    759 DELM_DSCR_LEN   =       3
                                017E    760
                                017E    761 INVRS_DSCR:                             ; Escape sequence used to turn on
   6D 37 5B 1B 00000186'010E0000'   017E    762             .ASCID  /[7m/               ;  inverse video attribute on a DECcrt.
                                018A    763
                                018A    764
                                018A    765 ;
                                018A    766 ; Misc messages and ascii strings.
                                018A    767 ;
                                018A    768
```

```
                                        018A   769 CLUSTER_MSG:
6F 20 77 65 69 56 00000192'010E0000'    018A   770         .ASCID  /View of Cluster from/
72 66 20 72 65 74 73 75 6C 43 20 66     0198
                              6D 6F     01A4
                                        01A6   771 LOCAL_MSG:
20 6C 61 63 6F 4C 000001AE'010E0000'    01A6   772         .ASCID  /Local ports on/
                 5E 6F 20 73 74 72 6F 70 01B4
                                        01BC   773 MSG1_DSCR:
65 74 73 7C 73 20 000001C4'010E0000'    01BC   774         .ASCID  / system ID /
                        20 44 49 20 6D  01CA
                                        01CF   775 MSG2_DSCR:
65 64 6F 6E 20 20 000001D7'010E0000'    01CF   776         .ASCID  /  node: /
                                 20 3A  01DD
                                        01DF   777 BUFF_MSG:
75 62 20 49 43 20 000001E7'010E0000'    01DF   778         .ASCID  / CI buffer descriptors in use./
70 69 72 63 73 65 64 20 72 65 66 66     01ED
2E 65 73 75 20 6E 69 20 73 72 6F 74     01F9
                                        0205   779 FREE_BUFF_MSG:
75 62 20 49 43 20 0000020D'010E0000'    0205   780         .ASCID  / CI buffer descriptors free./
70 69 72 63 73 65 64 20 72 65 66 66     0213
      2E 65 65 72 66 20 73 72 6F 74     021F
                                        0229   781
6E 61 6D 6D 6F 43 00000231'010E0000'    0229   782 PROMPT: .ASCID  /Command > /            ; Command input prompt string.
                        20 3E 20 64     0237
                           0000000A     023B   783 PROMPT_LEN = .-PROMPT-8
                                        023B   784
                                        023B   785
                                        023B   786 ;
                                        023B   787 ; Ascii strings for class headers.
                                        023B   788 ;
                                        023B   789
                                        023B   790 CLASS_NAMES:
            52 45 54 53 55 4C 43 00'    023B   791         .ASCIC  /CLUSTER/                ; Cluster (CLUB) class.
                                  07    023B
            53 4D 45 54 53 59 53 00'    0243   792         .ASCIC  /SYSTEMS/                ; Systems class.
                                  07    0243
            53 52 45 42 4D 45 4D 00'    024B   793         .ASCIC  /MEMBERS/                ; Members class.
                                  07    024B
         53 54 49 55 43 52 49 43 00'    0253   794         .ASCIC  /CIRCUITS/               ; Circuits class.
                                  08    0253
53 4E 4F 49 54 43 45 4E 4E 4F 43 00'    025C   795         .ASCIC  /CONNECTIONS/            ; Connections class.
                                  08    025C
         53 52 45 54 4E 55 4F 43 00'    0268   796         .ASCIC  /COUNTERS/               ; Counters class.
                                  08    0268
            53 54 49 44 45 52 43 00'    0271   797         .ASCIC  /CREDITS/                ; Credits class.
                                  07    0271
53 54 52 4F 50 5F 4C 41 43 4F 4C 00'    0279   798         .ASCIC  /LOCAL_PORTS/            ; Local-ports class.
                                  08    0279
               53 52 4F 52 52 45 00'    0285   799         .ASCIC  /ERRORS/                 ; Errors class.
                                  06    0285
                                        028C   800
                                        028C   801 ;
                                        02??   802 ; ASCIC strings used by display routines to convert an output value to an ascii
                                        0???   803 ; string.
                                        028C   804 ;
                                        028C   805
                                        028C   806 ;
```

```
                                    028C    807 ; Strings used by routine DISPLAY_CABLE_STATUS to display status of the paths
                                    028C    808 ; and cable (PB$B_CBL_STS, PB$B_P0_STS, and PB$B_P1_STS).
                                    028C    809 ;
                                    028C    810
                                    028C    811 CB_STAT_STRG:
                20 20 2D 20 20 00'  028C    812         .ASCIC  /  -  /                 ; Paths A and B are bad.
                                05  028C
                20 20 2D 20 41 00'  0292    813         .ASCIC  /A -  /                 ; Path A is good.
                                05  0292
                42 20 2D 20 20 00'  0298    814         .ASCIC  /  - B/                 ; Path B is good.
                                05  0298
                42 20 2D 20 41 00'  029E    815         .ASCIC  /A - B/                 ; Paths A and B are good.
                                05  029E
          44 45 53 53 4F 52 43 00'  02A4    816         .ASCIC  /CROSSED/               ; Cables are crossed.
                                07  02A4
                                    02AC    817
                          00000007  02AC    818 CB_STAT_STRNG_LEN = 7                   ; Maximum possible string length.
                                    02AC    819
                                    02AC    820 ;
                                    02AC    821 ; Strings used by DISPLAY_LOOPBACK_STATUS to display path loopback status
                                    02AC    822 ; (PDT$B_P0_LBSTS and PDT$B_P1_LBSTS).
                                    02AC    823 ;
                                    02AC    824
                                    02AC    825 LB_STAT_STRG:
                                    02AC    826
                42 20 2D 20 41 00'  02AC    827         .ASCIC  /A - B/                 ; Loopback tests passed on paths A and
                                05  02AC
                                    02B2    828                                         ;  B.
                20 20 2D 20 41 00'  02B2    829         .ASCIC  /A -  /                 ; Loopback tests passed on path A.
                                05  02B2
                42 20 2D 20 20 00'  02B8    830         .ASCIC  /  - B/                 ; Loopback tests passed on path B.
                                05  02B8
                20 20 2D 20 20 00'  02BE    831         .ASCIC  /  -  /                 ; Loopback tests failed on paths A and
                                05  02BE
                                    02C4    832                                         ;  B.
                20 41 2F 4E 20 00'  02C4    833         .ASCIC  @ N/A @                 ; Loopback testing is not being done.
                                05  02C4
                                    02CA    834
                          00000005  02CA    835 LB_STAT_STRG_LEN = 5                    ; Maximum possible string length.
                                    02CA    836
                                    02CA    837 ;
                                    02CA    838 ; Strings used by DISPLAY_RPORT_TYP to display the remote port type
                                    02CA    839 ; (PB$L_RPORT_TYP).
                                    02CA    840 ;
                                    02CA    841
                                    02CA    842 RPORT_TYP_STRNG:
                30 38 37 49 43 00'  02CA    843         .ASCIC  /CI780/                 ; Remote port is a CI780...
                                05  02CA
                30 35 37 49 43 00'  02D0    844         .ASCIC  /CI750/                 ; ... CI750.
                                05  02D0
                30 35 43 53 48 00'  02D6    845         .ASCIC  /HSC50/                 ; ... HSC50.
                                05  02D6
                                00' 02DC    846         .ASCIC  //                      ; ... Reserved.
                                00  02DC
                41 50 49 4C 4B 00'  02DD    847         .ASCIC  /KLIPA/                 ; ... KL10.
                                05  02DD
                   54 4E 49 43 00'  02E3    848         .ASCIC  /CINT/                  ; ... CI node tester.
```

```
                                                I 4
                     04  02E3
                49 4E 00' 02E8   849         .ASCIC  /NI/                    ; ... NI.
                     02  02E8
          48 54 53 53 41 50 00' 02EB   850         .ASCIC  /PASSTH/                ; ... Pass through mode.
                     06  02EB
                         02F2   851
                00000006 02F2   852 RPORT_TYP_STRNG_LEN = 6                 ; Maximum possible string length.
                         02F2   853
                         02F2   854 ;
                         02F2   855 ; Strings used by DISPLAY_PORT_TYPE to display the port type (UCB$B_DEVTYPE).
                         02F2   856 ;
                         02F2   857
                         02F2   858 DEV_TYP_STRNG:
             30 38 37 49 43 00' 02F2   859         .ASCIC  /C1780/                 ; Port is a C1780.
                     05  02F2
             30 35 37 49 43 00' 02F8   860         .ASCIC  /C1750/                 ; Port is a C1750.
                     05  02F8
                         02FE   861
                00000005 02FE   862 DEV_TYP_STRNG_LEN = 5                   ; Maximum possible string length.
                         02FE   863
                         02FE   864 ;
                         02FE   865 ; Strings used by DISPLAY_PORT_STATUS to display the port status (UCB$W_STS).
                         02FE   866 ;
                         02FE   867
                         02FE   868 PORT_STATUS_STRNG:
                         02FE   869
       45 4E 49 4C 46 46 4F 00' 02FE   870         .ASCIC  /OFFLINE/               ; Port is offline.
                     07  02FE
          45 4E 49 4C 4E 4F 00' 0306   871         .ASCIC  /ONLINE/                ; Port is online.
                     06  0306
                         030D   872
                00000007 030D   873 PORT_STATUS_STRNG_LEN = 7               ; Maximum possible string length.
                         030D   874
                         030D   875 ;
                         030D   876 ; Strings used by DISPLAY_CIR_STATE to display the state of a circuit
                         030D   877 ; (PB$W_STATE).
                         030D   878 ;
                         030D   879
                         030D   880 PB_STATE_STRNG:
       44 45 53 4F 4C 43 00' 030D   881         .ASCIC  /CLOSED/                ; Circuit is in the closed state.
                     06  030D
    54 4E 45 53 5F 54 53 00' 0314   882         .ASCIC  /ST_SENT/               ; Start sent.
                     07  0314
       43 45 52 5F 54 53 00' 031C   883         .ASCIC  /ST_REC/                ; Start received.
                     06  031C
             4E 45 50 4F 00' 0323   884         .ASCIC  /OPEN/                  ; Circuit is in the open state.
                     04  0323
    4C 49 41 46 5F 43 56 00' 0328   885         .ASCIC  /VC_FAIL/               ; Virtual circuit failure in progress.
                     07  0328
                         0330   886
                00000007 0330   887 PB_STATE_STRNG_LEN = 7                  ; Maximum possible string length.
                         0330   888
                         0330   889 ;
                         0330   890 ; Strings used by DISPLAY_RSTATE to display the state of the remote port
                         0330   891 ; (PB$B_RSTATE).
                         0330   892 ;
                         0330   893
```

```
                                0330   894 PB_RSTATE_STRNG:
          54 49 4E 49 4E 55 00' 0330   895         .ASCIC  /UNINIT/                ; Remote port is uninitialized.
                                06 0330
    54 49 4E 49 4E 55 5F 4D 00' 0337   896         .ASCIC  /M_UNINIT/              ; Remote port is in maintenance mode
                                08 0337
                                0340   897                                         ;  and is uninitialized.
             42 41 53 49 44 00' 0340   898         .ASCIC  /DISAB/                 ; Remote port is disabled.
                                05 0340
       42 41 53 49 44 5F 4D 00' 0346   899         .ASCIC  /M_DISAB/               ; Remote port is in maintenance mode
                                07 0346
                                034E   900                                         ;  and is disabled.
                42 41 4E 45 00' 034E   901         .ASCIC  /ENAB/                  ; Remote port is enabled.
                                04 034E
          42 41 4E 45 5F 4D 00' 0353   902         .ASCIC  /M_ENAB/                ; Remote port is in maintenance mode
                                06 0353
                                035A   903                                         ;  and is enabled.
                                035A   904
                    00000008    035A   905 PB_RSTATE_STRNG_LEN = 8                 ; Maximum possible string length.
                                035A   906
                                035A   907 ;
                                035A   908 ; Strings used by DISPLAY_CON_STATE to display the connection state
                                035A   909 ; (CDT$W_STATE).
                                035A   910 ;
                                035A   911
                                035A   912 CDT_STATE_STRNG:
          44 45 53 4F 4C 43 00' 035A   913         .ASCIC  /CLOSED/                ; Connection is in the closed state.
                                06 035A
          4E 45 54 53 49 4C 00' 0361   914         .ASCIC  /LISTEN/                ; Connection is in the listen state.
                                06 0361
                4E 45 50 4F 00' 0368   915         .ASCIC  /OPEN/                  ; Connection is in the open state.
                                04 0368
       4B 43 41 5F 43 53 49 44 00' 036D   916         .ASCIC  /DISC_ACK/              ; Disconnect acknowledged.
                                08 036D
       43 45 52 5F 43 53 49 44 00' 0376   917         .ASCIC  /DISC_REC/              ; Disconnect request received.
                                08 0376
    54 4E 45 53 5F 43 53 49 44 00' 037F   918         .ASCIC  /DISC_SENT/             ; Disconnect sent.
                                09 037F
    48 43 54 4D 5F 43 53 49 44 00' 0389   919         .ASCIC  /DISC_MTCH/             ; Disconnect match.
                                09 0389
       54 4E 45 53 5F 4E 4F 43 00' 0393   920         .ASCIC  /CON_SENT/              ; Connect request sent.
                                08 0393
          4B 43 41 5F 4E 4F 43 00' 039C   921         .ASCIC  /CON_ACK/               ; Connect request sent & acknowledged.
                                07 039C
          43 45 52 5F 4E 4F 43 00' 03A4   922         .ASCIC  /CON_REC/               ; Connect request received.
                                07 03A4
    54 4E 45 53 5F 50 43 43 41 00' 03AC   923         .ASCIC  /ACCP_SENT/             ; Accept request sent.
                                09 03AC
       54 4E 45 53 5F 4A 45 52 00' 03B6   924         .ASCIC  /REJ_SENT/              ; Reject sent.
                                08 03B6
          4C 49 41 46 5F 43 56 00' 03BF   925         .ASCIC  /VC_FAIL/               ; Virtual circuit failed.
                                07 03BF
                                03C7   926
                    00000009    03C7   927 DISPLAY_CON_STATE_LEN = 9                ; Maximum possible string length.
                                03C7   928
                                03C7   929 ;
                                03C7   930 ; Strings used by DISPLAY_SCS_STATE to display the SCS send blocked state
                                03C7   931 ; (CDT$W_BLKSTATE).
```

```
                                03C7   932 ;
                                03C7   933
                                03C7   934 CDT_BLKSTATE_STRNG:
                52 41 45 4C 43 00' 03C7   935         .ASCIC  /CLEAR/                 ; Not blocked.
                                   05  03C7
           44 4E 45 50 5F 4E 4F 43 00' 03CD   936         .ASCIC  /CON_PEND/              ; Waiting to send connection request.
                                   08  03CD
        44 4E 45 50 5F 50 43 43 41 00' 03D6   937         .ASCIC  /ACCP_PEND/             ; Waiting to send accept request.
                                   09  03D6
           44 4E 45 50 5F 4A 45 52 00' 03E0   938         .ASCIC  /REJ_PEND/              ; Waiting to send reject request.
                                   08  03E0
        44 4E 45 50 5F 43 53 49 44 00' 03E9   939         .ASCIC  /DISC_PEND/             ; Waiting to send disconnect request.
                                   09  03E9
           44 4E 45 50 5F 52 43 00' 03F3   940         .ASCIC  /CR_PEND/               ; Waiting to send credit.
                                   07  03F3
           44 4E 45 50 5F 52 43 44 00' 03FB   941         .ASCIC  /DCR_PEND/              ; Waiting to send credit in
                                   08  03FB
                                0404   942                                             ;  preparation for disconnect.
                                0404   943
                        00000009  0404   944 DISPLAY_SCS_STATE_LEN = 9                 ; Maximum possible string length.
                                0404   945
                                0404   946 ;
                                0404   947 ; Strings used to display quorum file states.
                                0404   948 ;
                                0404   949
                                0404   950 YES_NO_STRING:
                                0404   951
                     53 45 59 00' 0404   952         .ASCIC  /YES/
                                   03  0404
                        4F 4E 00' 0408   953         .ASCIC  /NO/
                                   02  0408
                                040B   954
                        00000003  040B   955 YES_NO_STRNG_LEN = 3                     ; Maximum length of string.
                                040B   956
                                040B   957 ;
                                040B   958 ; Strings used by DISPLAY_CSB_STATE to display the cluster system block state
                                040B   959 ; (CSB$B_STATE).
                                040B   960 ;
                                040B   961
                                040B   962 CSB_STATE_STRNG:
                     4E 45 50 4F 00' 040B   963         .ASCIC  /OPEN/                  ; Open.
                                   04  040B
                  44 45 53 4F 4C 43 00' 0410   964         .ASCIC  /CLOSED/                ; Closed.
                                   06  0410
        54 43 45 4E 4E 4F 43 45 52 00' 0417   965         .ASCIC  /RECONNECT/             ; Attempting to reconnect.
                                   09  0417
                        57 45 4E 00' 0421   966         .ASCIC  /NEW/                   ; New block.
                                   03  0421
              54 43 45 4E 4E 4F 43 00' 0425   967         .ASCIC  /CONNECT/               ; Attempting initial connection.
                                   07  0425
                 54 50 45 43 43 41 00' 042D   968         .ASCIC  /ACCEPT/                ; Accepting initial connection.
                                   06  042D
     54 43 45 4E 4E 4F 43 53 49 44 00' 0434   969         .ASCIC  /DISCONNECT/            ; Disconnect in progress.
                                   0A  0434
           54 50 45 43 43 41 45 52 00' 043F   970         .ASCIC  /REACCEPT/              ; Accepting reconnect request.
                                   08  043F
                     54 49 41 57 00' 0448   971         .ASCIC  /WAIT/                  ; Time-out in progress.
```

```
                     04  0448
          44 41 45 44 00' 044D   972          .ASCIC  /DEAD/                  ; No connection possible.
                     04  044D
                         0452   973
                0000000A 0452   974 CSB_STATE_STRNG_LEN = 10                 ; Maximum possible length of string.
                         0452   975
                         0452   976 ;
                         0452   977 ; Strings used by DISPLAY_CSB_STATUS to display the status of the node in the
                         0452   978 ; cluster (CSB$L_STATUS).
                         0452   979 ;
                         0452   980
                         0452   981 CSB_STATUS_STRNG:
             57 45 4E 00' 0452   982          .ASCIC  /NEW/                   ; New system in cluster.
                     03  0452
       57 45 4E 5F 4B 52 42 00' 0456   983          .ASCIC  /BRK_NEW/               ; New system - break in connection.
                     07  0456
          52 45 42 4D 45 4D 00' 045E   984          .ASCIC  /MEMBER/                ; System is a member of the cluster.
                     06  045E
       4D 45 4D 5F 4B 52 42 00' 0465   985          .ASCIC  /BRK_MEM/               ; Member - break in connection.
                     07  0465
             4E 4F 4E 00' 046D   9            .ASCIC  /NON/                   ; System is not a member of the cluster.
                     03  046D
       4E 4F 4E 5F 4B 52 42 00' 0471   987          .ASCIC  /BRK_NON/               ; Non-member - break in connection.
                     07  0471
       44 45 56 4F 4D 45 52 00' 0479   988          .ASCIC  /REMOVED/               ; System is removed from the cluster.
                     07  0479
       4D 45 52 5F 4B 52 42 00' 0481   989          .ASCIC  /BRK_REM/               ; Removed system - break in connection.
                     07  0481
                         0489   990
                00000007 0489   991 CSB_STATUS_STRNG_LEN = 7                 ; Maximum possible string length.
                         0489   992
                         0489   993 ;
                         0489   994 ; Symbol prefix strings used by DISPLAY_FIELD to determine which SCS data
                         0489   995 ; structure contains the data.  Prefixes must be less than 4 characters.
                         0489   996 ;
                         0489   997
                         0489   998 SYM_PRFX:
       42 55 4C 43 00' 0489   999          .ASCIC  /CLUB/                  ; Prefix for club offsets.
                     04  0489
          24 42 53 00' 048E  1000          .ASCIC  /SB$/                   ; Prefix for system block offsets.
                     03  048E
       24 42 53 43 00' 0492  1001          .ASCIC  /CSB$/                  ; Prefix for cluster system block
                     04  0492
                         0497  1002                                          ;  offsets.
          24 42 50 00' 0497  1003          .ASCIC  /PB$/                   ; Prefix for path block offsets.
                     03  0497
       24 54 44 43 00' 049B  1004 CDT_PFX:.ASCIC  /CDT$/                  ; Prefix for connection descriptor
                     04  049B
                         04A0  1005                                          ;  table offsets.
       24 54 44 50 00' 04A0  1006 PDT_PFX:.ASCIC  /PDT$/                  ; Prefix for port descriptor table
                     04  04A0
                         04A5  1007                                          ;  offsets.
          24 42 4D 00' 04A5  1008          .ASCIC  /MB$/                   ; Prefix for miscellaneous buffer
                     03  04A5
                         04A9  1009                                          ;  offsets.
                         04A9  1010
                00000007 04A9  1011 PFX_LIM = 7                             ; Number of valid prefixes.
```

```
                        04A9  1012
                        04A9  1013
                        04A9  1014 ;
                        04A9  1015 ; Table used to convert SCS data size in bytes (FD$B_LENGTH) to number of
                        04A9  1016 ; columns needed to display the data (FD$B_MAX_DATA_SIZE).
                        04A9  1017 ;
                        04A9  1018
                        04A9  1019 CVT_TABLE:
14 11 0F 0D 0A 08 05 03 00  04A9  1020          .BYTE   0,3,5,8,10,13,15,17,20
                        04B2  1021
                        04B2  1022 ;
                        04B2  1023 ; Table used to convert SB$L_HW_TYPE into a bit value to determine if
                        04B2  1024 ; the type of system should be displayed.
                        04B2  1025 ;
                        04B2  1026
                        04B2  1027 ASSUME  CLS$V_750_SYS    EQ      16
                        04B2  1028 ASSUME  CLS$V_780_SYS    EQ      CLS$V_750_SYS+1
                        04B2  1029 ASSUME  CLS$V_785_SYS    EQ      CLS$V_780_SYS+1
                        04B2  1030 ASSUME  CLS$V_790_SYS    EQ      CLS$V_785_SYS+1
                        04B2  1031 ASSUME  CLS$V_HSC50_SYS  EQ      CLS$V_790_SYS+1
                        04B2  1032 ASSUME  CLS$V_HSC60_SYS  EQ      CLS$V_HSC50_SYS+1
                        04B2  1033
                        04B2  1034 HW_TYPE_TABLE:
            30353756    04B2  1035          .LONG   ^A/V750/                ; 750 systems.
            30383756    04B6  1036          .LONG   ^A/V780/                ; 780 systems.
            35383756    04BA  1037          .LONG   ^A/V785/                ; 785 systems.
            30303838    04BE  1038          .LONG   ^A/8800/                ; 790 systems.
            30355348    04C2  1039          .LONG   ^A/HS50/                ; HSC50 systems.
            30365348    04C6  1040          .LONG   ^A/HS60/                ; HSC60 systems.
            00000006    04CA  1041 NUM_HW_TYPES =  <.-HW_TYPE_TABLE>/4
                        04CA  1042
                        04CA  1043 ;
                        04CA  1044 ; Table used for Hex to decimal conversion.
                        04CA  1045 ;
                        04CA  1046
8AC72304 89E8C000       04CA  1047 DEC_TAB:.QUAD    10000000000000000000
0DE0B6B3 A7640000       04D2  1048          .QUAD   1000000000000000000
01634578 5D8A0000       04DA  1049          .QUAD   100000000000000000
002386F2 6FC10000       04E2  1050          .QUAD   10000000000000000
00038D7E A4C68000       04EA  1051          .QUAD   1000000000000000
00005AF3 107A4000       04F2  1052          .QUAD   100000000000000
00000918 4E72A000       04FA  1053          .QUAD   10000000000000
000000E8 D4A51000       0502  1054          .QUAD   1000000000000
00000017 4876E800       050A  1055          .QUAD   100000000000
00000002 540BE400       0512  1056          .QUAD   10000000000
00000000 3B9ACA00       051A  1057          .QUAD   1000000000
00000000 05F5E100       0522  1058          .QUAD   100000000
00000000 00989680       052A  1059          .QUAD   10000000
00000000 000F4240       0532  1060          .QUAD   1000000
00000000 000186A0       053A  1061          .QUAD   100000
00000000 00002710       0542  1062          .QUAD   10000
00000000 000003E8       054A  1063          .QUAD   1000
00000000 00000064       0552  1064          .QUAD   100
00000000 0000000A       055A  1065          .QUAD   10
00000000 00000001       0562  1066          .QUAD   1
                        056A  1067
            00000014    056A  1068 DEC_TAB_LEN = .-DEC_TAB/8
```

```
                     056A  1069 DEC_TAB_END:
                     056A  1070
                     056A  1071 ;
                     056A  1072 ; Starting and ending addresses of pages to be locked in working set.
                     056A  1073 ; This includes the fixed length snapshot buffers and the snapshot routine
                     056A  1074 ; itself.  It does not include the variable length buffers which must be
                     056A  1075 ; locked separately.
                     056A  1076 ;
                     056A  1077
                     056A  1078 SNAP_ADRS:
                     056A  1079         .ADDRESS -                      ; Starting and ending addresses of
00003333'00000000'   056A  1080                 SNAP_BUF,SNAP_END       ;  snapshot routine.
                     0572  1081
                     0572  1082 ;
                     0572  1083 ; Item list for terminal QIO function modifiers.
                     0572  1084 ;
                     0572  1085
                     0572  1086 TERM_LIST:
                0000 0572  1087         .WORD   0
                0000 0574  1088         .WORD   TRM$_MODIFIERS
            00008000 0576  1089         .LONG   TRM$M_TM_NOEDIT
            00000000 057A  1090         .LONG   0
            0000000C 057E  1091 TERM_LIST_LEN = .-TERM_LIST
                     057E  1092
                     057E  1093 ;
                     057E  1094 ; List of parameters request by $GETDVI.
                     057E  1095 ;
                     057E  1096
                     057E  1097 GET_LIST:
                0004 057E  1098         .WORD   4                       ; Buffer size.
                0004 0580  1099         .WORD   DVI$_DEVCLASS           ; Request device class
                     0582  1100         .ADDRESS -                      ; Buffer address.
00002208'            0582  1101                 DEVICE_CLASS            ;
00000000             0586  1102         .LONG   0                       ; Length not needed.
                     058A  1103
                0004 058A  1104         .WORD   4                       ; Buffer size.
                0006 058C  1105         .WORD   DVI$_DEVTYPE            ; Request device type.
                     058E  1106         .ADDRESS -                      ; Buffer address.
0000220C'            058E  1107                 DEVICE_TYPE             ;
00000000             0592  1108         .LONG   0                       ; Length not needed.
                     0596  1109
                0004 0596  1110         .WORD   4                       ; Buffer size.
                000A 0598  1111         .WORD   DVI$_DEVDEPEND          ; Request device dependent information.
                     059A  1112         .ADDRESS -                      ; Buffer address.
00002218'            059A  1113                 DEVDEPEND               ;
00000000             059E  1114         .LONG   0                       ; Length not needed.
                     05A2  1115
                0004 05A2  1116         .WORD   4                       ; Buffer size.
                001C 05A4  1117         .WORD   DVI$_DEVDEPEND2         ; Request additional device dependent
                     05A6  1118                                         ;  information.
                     05A6  1119         .ADDRESS -                      ; Buffer address.
0000221C'            05A6  1120                 DEVDEPEND2              ;
00000000             05AA  1121         .LONG   0                       ; Length not needed.
                     05AE  1122
                0004 05AE  1123         .WORD   4                       ; Buffer size.
                0008 05B0  1124         .WORD   DVI$_DEVBUFSIZ          ; Request device buffer size.
                     05B2  1125         .ADDRESS -                      ; Buffer address.
```

```
                                                B 5
00002210' 05B2  1126                    BUFFER_SIZE
00000000  05B6  1127            .LONG   0                       ; Length not needed.
          05BA  1128
00000000  05BA  1129            .LONG   0                       ; End of list.
          05BE  1130
          05BE  1131
          05BE  1132 ;
          05BE  1133 ; List of parameters request by $GETJPI.
          05BE  1134 ;
          05BE  1135
          05BE  1136 JPI_LIST:
    0004  05BE  1137            .WORD   4                       ; Buffer size.
    0000' 05C0  1138            .WORD   JPI$_CURPRIV            ; Request process's current privileges.
          05C2  1139            .ADDRESS -                      ; Buffer address.
00002230' 05C2  1140                    CUR_PRIV                ;
00000000  05C6  1141            .LONG   0                       ; Length not needed.
          05CA  1142
00000000  05CA  1143            .LONG   0                       ; End of list.
          05CE  1144
          05CE  1145
          05CE  1146            .SBTTL  Local storage.
```

```
                00000000  1148          .PSECT  RWDATA,RD,WRT,NOEXE
                    0000  1149
           00000000 0000  1150 BUFADR:  .LONG   0                       ; Temporary storage for queue buffer
                    0004  1151                                          ;  address.
           00000100' 0004  1152 BUFSIZE: .LONG   MBX_BUF_LEN            ; Buffer size.
                    0008  1153
                    0008  1154 MSG_FRE_QUE:                             ; Queue of allocated message buffers.
00000008'00000008'  0008  1155          .LONG   MSG_FRE_QUE,MSG_FRE_QUE ;
                    0010  1156 MSG_OUT_QUE:                             ; Queue of broadcast messages to be
00000010'00000010'  0010  1157          .LONG   MSG_OUT_QUE,MSG_OUT_QUE ;  output.
                    0018  1158
                    0018  1159 SCRATCH:                                 ; Temporary storage used for displaying
           00000000 0018  1160          .LONG   0                       ;  PROTOCOL field.
                    001C  1161
                    001C  1162 ;
                    001C  1163 ; File access block for input file.  Used when executing a command procedure
                    001C  1164 ; or when executing Show Cluster from a file stream.  The default filename
                    001C  1165 ; extention is .COM.
                    001C  1166 ;
                    001C  1167
                    001C  1168 INFAB:   $FAB    DNM = <.COM>,-
                    001C  1169                  FNA = IN_FILE_SPEC,-
                    001C  1170                  FNS = IN_FILE_SPEC_LEN,-
                    001C  1171                  FOP = NAM,-
                    001C  1172                  NAM = INAM_BLOCK
                    006C  1173
                    006C  1174 INAM_BLOCK:
                    006C  1175          $NAM    ESA = IN_EXP_NAME,-     ; Address to store expanded filespec.
                    006C  1176                  ESS = IN_EXP_NAME_LEN   ; Address to store length of expanded
                    00CC  1177                                          ;  filespec.
                    00CC  1178
                    00CC  1179 IN_FILE_SPEC:                            ; Buffer for filespec for procedure.
           0000011C 00CC  1180          .BLKB   80                      ;
           00000050 011C  1181 IN_FILE_SPEC_LEN = .-IN_FILE_SPEC       ;
                    011C  1182
                    011C  1183 IN_EXP_DSCR:                             ; Descriptor for expanded filespec.
           00000050 011C  1184          .LONG   80                      ;
           00000124' 0120  1185          .LONG   IN_EXP_NAME             ;
                    0124  1186 IN_EXP_NAME:                             ; Buffer for expanded filespec.
           00000174 0124  1187          .BLKB   80                      ;
           00000050 0174  1188 IN_EXP_NAME_LEN = . -IN_EXP_NAME         ;
                    0174  1189
                    0174  1190
                    0174  1191 ;
                    0174  1192 ; Output file access block.  Used for /OUTPUT= or when SYS$OUTPUT has been
                    0174  1193 ; defined to be a file.  The default filename is SHWCLSTR.DAT.
                    0174  1194 ;
                    0174  1195
                    0174  1196 OUTFAB: $FAB    DNM = <SHWCLSTR.DAT>,-
                    0174  1197                  FAC = PUT,-
                    0174  1198                  FNA = OUT_FILE_SPEC,-
                    0174  1199                  FNS = OUT_FILE_SPEC_LEN,-
                    0174  1200                  FOP = NAM,-
                    0174  1201                  NAM = OUTNAM_BLOCK
                    01C4  1202
                    01C4  1203 OUTNAM_BLOCK:
                    01C4  1204          $NAM    ESA = OUT_EXP_NAME,-    ; Address to store expanded filespec.
```

```
                01C4  1205                      ESS = OUT_EXP_NAME_LEN  ; Address to store length of expanded
                0224  1206                                              ;  filespec.
                0224  1207
                0224  1208 OUT_FILE_SPEC:                               ; Buffer for filespec for procedure.
00000274        0224  1209                .BLKB   80                    ;
00000050        0274  1210 OUT_FILE_SPEC_LEN = .-OUT_FILE_SPEC          ;
                0274  1211
                0274  1212 OUT_EXP_DSCR:                                ; Descriptor for expanded filespec.
00000050        0274  1213                .LONG   80                    ;
0000027C'       0278  1214                .LONG   OUT_EXP_NAME          ;
                027C  1215 OUT_EXP_NAME:                                ; Buffer for expanded filespec.
000002CC        027C  1216                .BLKB   80                    ;
00000050        02CC  1217 OUT_EXP_NAME_LEN = . -OUT_EXP_NAME           ;
                02CC  1218
                02CC  1219 ;
                02CC  1220 ; Establish a record access block.  Used when inputing from a procedure or
                02CC  1221 ; when executing from a file stream.
                02CC  1222 ;
                02CC  1223
                02CC  1224 INRAB:  $RAB     FAB = INFAB,-               ; Pointer to FAB.
                02CC  1225                  UBF = REC_BUFF,-            ; Record buffer.
                02CC  1226                  USZ = REC_SIZE              ; Record buffer size.
                0310  1227
                0310  1228 REC_DSCR:                                    ; Descriptor for record buffer.
00000000        0310  1229                .LONG   0                     ;
00000318'       0314  1230                .LONG   REC_BUFF              ;
                0318  1231 REC_BUFF:                                    ; Record buffer.
0000039C        0318  1232                .BLKB   132                   ;
00000084        039C  1233 REC_SIZE = .-REC_BUFF                        ;
                039C  1234
                039C  1235
                039C  1236 TRAN_DSCR:                                   ; Descriptor for SHOW_CLUSTER$INIT
                039C  1237                                              ;  translation.
00000040        039C  1238                .LONG   64                    ; Length.
                03A0  1239                .ADDRESS -                    ; Address of buffer.
000003A4'       03A0  1240                         TRAN_BUF             ;
                03A4  1241 TRAN_BUF:                                    ; Buffer.
000003E4        03A4  1242                .BLKB   64                    ;
                03E4  1243
                03E4  1244 BUF_COUNT:                                   ; Number of times we've allocated some
      00        03E4  1245                .BYTE   0                     ;  buffer space. (2 max)
                03E5  1246
                03E5  1247 SB_COUNT:                                    ; Maximum number of entries to be stored
    0000        03E5  1248                .WORD   0                     ;  in the system block buffer.
                03E7  1249 CSB_COUNT:                                   ; Maximum number of entries to be stored
    0000        03E7  1250                .WORD   0                     ;  in the cluster system block buffer.
                03E9  1251 PB_COUNT:                                    ; Maximum number of entries to be stored
    0000        03E9  1252                .WORD   0                     ;  in the path block buffer.
                03EB  1253 CDT_COUNT:                                   ; Maximum number of entries to be stored
    0000        03EB  1254                .WORD   0                     ;  in the CDT buffer.
                03ED  1255
00000000        03ED  1256 VAR_BUF:.LONG    0                           ; Address of variable length buffer
                03F1  1257                                              ;  returned from GET_VM.
                03F1  1258 VAR_BUF_END:                                 ; End of variable buffer space.
00000000        03F1  1259                .LONG   0                     ;
                03F5  1260 VAR_BUF_LEN:                                 ; Length of variable buffer.
00000000        03F5  1261                .LONG   0                     ;
```

```
                     03F9  1262
                     03F9  1263 ;
                     03F9  1264 ; Descriptors for variable length buffers from previous snapshot.
                     03F9  1265 ;
                     03F9  1266
                     03F9  1267 OLD_SB_DSCR:                            ; System block buffer.
 00000000 00000000   03F9  1268         .QUAD   0                      ;
                     0401  1269 OLD_CSB_DSCR:                           ; Cluster system block buffer.
 00000000 00000000   0401  1270         .QUAD   0                      ;
                     0409  1271 OLD_PB_DSCR:                            ; Path block buffer.
 00000000 00000000   0409  1272         .QUAD   0                      ;
                     0411  1273 OLD_CDT_DSCR:                           ; CDT buffer.
 00000000 00000000   0411  1274         .QUAD   0                      ;
                     0419  1275
                     0419  1276 OLD_VAR_BUF:                           ; Address of previous buffer.
         00000000    0419  1277         .LONG   0                      ;
                     041D  1278
                     041D  1279 OLD_VAR_BUF_END:                       ; End of previous buffer.
         00000000    041D  1280         .LONG   0                      ;
                     0421  1281
                     0421  1282 OLD_VAR_BUF_LEN:                       ; Length of previous buffer.
         00000000    0421  1283         .LONG   0                      ;
                     0425  1284
                     0425  1285 NUM_FIELDS:                            ; Number of fields to be displayed
               00    0425  1286         .BYTE   0                      ;  Used to limit left-scrolling.
                     0426  1287 SCS_NUM_FIELDS:                        ; Number of SCS fields to be displayed.
               00    0426  1288         .BYTE   0                      ;
                     0427  1289 CLUB_NUM_FIELDS:                       ; Number of CLUB fields to be displayed.
               00    0427  1290         .BYTE   0                      ;
                     0428  1291
                     0428  1292 NUM_LINES:                             ; Number of lines of data this pass.
         00000000    0428  1293         .LONG   0                      ;
                     042C  1294 PRIOR_NUM_LINES:                       ; Number of lines of data last pass.
         00000000    042C  1295         .LONG   0                      ;
                     0430  1296 SCROLL_LINES:                          ; Copy of PRIOR_NUM_LINES used to
         00000000    0430  1297         .LONG   0                      ;  limit up scrolling.
                     0434  1298
                     0434  1299 ;
                     0434  1300 ; LINE_DIF is a record of the lines in the display that need to be redrawn
                     0434  1301 ; because data on that line has changed.  Bit 7 corresponds to line #7,
                     0434  1302 ; Bit 8 - line #8, etc.  LINE_DIF is used to save the time of redrawing lines
                     0434  1303 ; in the display that haven't changed since the last pass.
                     0434  1304 ;
                     0434  1305 ; PRIOR_LINE_DIF is a record of the lines that were redrawn in the last pass.
                     0434  1306 ; PRIOR_LINE_DIF is used to turn off fields that have been set to inverse
                     0434  1307 ; video on the previous pass.
                     0434  1308 ;
                     0434  1309 ; Differences in lines are detected by the following routines:
                     0434  1310 ;
                     0434  1311 ;       GET_DATA        (Detects changes in data through setting of inverse
                     0434  1312 ;                       video flag by the CHECK_DIF routine)
                     0434  1313 ;       CHECK_POS       Detects change in area of the screen where a system
                     0434  1314 ;                       is displayed without the data changing.
                     0434  1315 ;       CHECK_NUM_PB    Detects changes in the number of circuits so that
                     0434  1316 ;                       NUM_CIRCUITS is updated.
                     0434  1317 ;       CHECK_NUM_CDT   Detects changes in the number of connections so that
                     0434  1318 ;                       NUM_CONNECTIONS is updated.
```

```
                     0434  1319 ;
                     0434  1320 ; NOTES:
                     0434  1321 ;
                     0434  1322 ;       LINE_DIF is cleared (actually SCROLL_DIF is copied to LINE_DIF)
                     0434  1323 ;       before each display is built after being copied to PRIOR_LINE_DIF.
                     0434  1324 ;
                     0434  1325 ;       LINE_DIF is used only by GET_DATA to determine if the line has to
                     0434  1326 ;       be redrawn because the line has fields that need to be updated.
                     0434  1327 ;
                     0434  1328 ;       PRIOR_LINE_DIF is used only by GET_DATA to redrawn the line so that
                     0434  1329 ;       any fields set to inverse video will be cleared.
                     0434  1330 ;
                     0434  1331
                     0434  1332 LINE_DIF:                                  ; Reserve one longword to record the
00000000             0434  1333         .LONG   0                          ; changes in data lines on the current
                     0438  1334                                            ;  pass.
                     0438  1335 PRIOR_LINE_DIF:                            ; Reserve one longword to save a record
00000000             0438  1336         .LONG   0                          ; of changes in data lines on the
                     043C  1337                                            ;  previous pass.
                     043C  1338
                     043C  1339 ;
                     043C  1340 ; BLANKS_LINES is used to record which lines in the display have all the fields
                     043C  1341 ; filled in with blanks.  This together with PRIOR_BLANK_LINES is used to
                     043C  1342 ; save the time of redrawing a blank line every time the screen is updated.
                     043C  1343 ;
                     043C  1344 ;NOTES:
                     043C  1345 ;
                     043C  1346 ;       BLANK_LINES is cleared before each display is build after being
                     043C  1347 ;       copied to PRIOR_BLANK_LINES.
                     043C  1348 ;
                     043C  1349 ;       BLANK_LINES and PRIOR_BLANK_LINES are used only by the GET_BLANK
                     043C  1350 ;       routine to determine if the line should be drawn or not.
                     043C  1351 ;
                     043C  1352
                     043C  1353 BLANK_LINES:                                ; Reserve one longword to save a record
00000000             043C  1354         .LONG   0                          ;  of the blank lines that are output.
                     0440  1355 PRIOR_BLANK_LINES:                         ; Reserve one longword to save a record
00000000             0440  1356         .LONG   0                          ;  of blank lines from previous pass.
                     0444  1357
                     0444  1358 ;
                     0444  1359 ; SCROLL_DIF is used to record which lines is the display must be redrawn
                     0444  1360 ; because the screen was hardware scrolled.  When the screen is hardware
                     0444  1361 ; scrolled one or more blank lines are left at the top or bottom of the
                     0444  1362 ; scrolling region and need to be filled in.
                     0444  1363 ;
                     0444  1364 ;NOTES:
                     0444  1365 ;
                     0444  1366 ;       SCROLL_DIF is cleared after the display has been built and before
                     0444  1367 ;       any scroll commands can be issued.
                     0444  1368 ;
                     0444  1369 ;       SCROLL_DIF is copied to LINE_DIF before the display is built so
                     0444  1370 ;       that the line are considered "changed" lines.
                     0444  1371 ;
                     0444  1372
                     0444  1373 SCROLL_DIF:                                 ; Indicates which line on a DECcrt
00000000             0444  1374         .LONG   0                          ;  must be redrawn because the screen
                     0448  1375                                            ;  scrolled.  (Because the screen is
```

```
                     0448  1376                                    ;  hardware scrolled).
                     0448  1377
                     0448  1378 NODE_NAME_DSCR:                    ; Reserve room for storing the node
00000000             0448  1379          .LONG   0                 ;  name of the local system.
00000450'            044C  1380          .LONG   NODE_NAME         ;
                     0450  1381 NODE_NAME:                        ;
00000458             0450  1382          .BLKB   8                 ;
                     0458  1383
                     0458  1384 ;
                     0458  1385 ; Scrolling parameters.
                     0458  1386 ;
                     0458  1387
                     0458  1388 SCROLL_DELAY:                     ; Number of refreshes it takes to
00000000             0458  1389          .LONG   0                 ;  scroll the bottom two lines out of
                     045C  1390                                    ;  the scrolling region.
                     045C  1391
00000000             045C  1392 V_SCROLL:.LONG  0                  ; Number of data lines to be skipped by
                     0460  1393                                    ;  the GET_DATA routine before a line
                     0460  1394                                    ;  of data is added to the screen
                     0460  1395                                    ;  buffer.
                     0460  1396 V_SCROLL_LINE:                    ; Counter for the number of lines
00000000             0460  1397          .LONG   0                 ;  skipped by the GET_DATA routine.
                     0464  1398
                     0464  1399 CLSTR_H_SCROLL:                   ; Number of field entries to be skipped
00000000             0464  1400          .LONG   0                 ;  in the cluster report by the
                     0468  1401                                    ;  FIND_ENTRY routine before success is
                     0468  1402                                    ;   returned.
                     0468  1403
                     0468  1404 LOCAL_H_SCROLL:                   ; Number of field entries to be skipped
00000000             0468  1405          .LONG   0                 ;  in the local report by the
                     046C  1406                                    ;  FIND_ENTRY routine before success is
                     046C  1407                                    ;  returned.
                     046C  1408
                     046C  1409 H_SCROLL_LINE:                    ; Counter for the number of entries
00000000             046C  1410          .LONG   0                 ;  skipped by the FIND_ENTRY routine.
                     0470  1411
                     0470  1412 TOP_CMD_LINE:                     ; Indicates type of information last
00                   0470  1413          .BYTE   0                 ;  output on top command line.
                     0471  1414
                     0471  1415 BOT_CMD_LINE:                     ; Indicates type of information last
00                   0471  1416          .BYTE   0                 ;  output on bottom command line.
                     0472  1417
                     0472  1418
                     0472  1419 ;
                     0472  1420 ; Flags.
                     0472  1421 ;
                     0472  1422
                     0472  1423 FLAG_STATE:                       ; Longword to receive event flag
00000000             0472  1424          .LONG   0                 ;  status.
                     0476  1425
                     0476  1426 MISC_FLAGS:                       ; Miscellaneous flags.
00000000             0476  1427          .LONG   0                 ;
                     047A  1428
                     047A  1429 SCROLL_FLAGS:                     ; Flags to indicate which way the
00                   047A  1430          .BYTE   0                 ;  screen can be scrolled.
                     047B  1431
                     047B  1432 LOCAL_SCROLL_FLAGS:               ; Used to save a copy of the scroll
```

```
                     00  047B  1433          .BYTE   0                       ;  flags for the local report.
                         047C  1434
                         047C  1435 CLSTR_SCROLL_FLAGS:                      ; Used to save a copy of the scroll
                     00  047C  1436          .BYTE   0                       ;  flags for the cluster report.
                         047D  1437
                         047D  1438 HELP_FLAGS:                              ; Flags for LBR$OUTPUT_HELP.
             00000021    047D  1439          .LONG   HLP$M_PROMPT!HLP$M_HELP ; Run help in prompting mode.
                         0481  1440                                          ; With list of topics available.
                         0481  1441
                         0481  1442 ;
                         0481  1443 ; Exit routine.
                         0481  1444 ;
                         0481  1445
                         0481  1446 EXIT_BLK:
             00000000    0481  1447          .LONG   0                       ; Forward link.
                         0485  1448          .ADDRESS -                      ; Exit handler address.
             00000F39'   0485  1449                  EXIT_RTN                ;
             00000001    0489  1450          .LONG   1                       ; Number of arguments.
                         048D  1451          .ADDRESS -                      ; Address to receive exit status.
             00000491'   048D  1452                  STATUS                  ;
                         0491  1453
                         0491  1454 STATUS:                                  ; Status code from $EXIT.
             00000000    0491  1455          .LONG   0                       ;
                         0495  1456
                         0495  1457 ;
                         0495  1458 ; Descriptor for token.  Used when a class name is parsed in case an
                         0495  1459 ; error occurs.
                         0495  1460 ;
                         0495  1461
                         0495  1462 TKN_DSCR:
00000000 00000000        0495  1463          .QUAD   0
                         049D  1464
                         049D  1465
                         049D  1466 ERR_MSGVEC:
                 0001    049D  1467          .WORD   1                       ; Argument count.
                 000F    049F  1468          .WORD   ^X0F                    ; Message flags.
             00000000    04A1  1469          .LONG   0                       ; Message identification.
                 0001    04A5  1470          .WORD   1                       ; FAO count.
                 0000    04A7  1471          .WORD   0                       ; New message flags.
                         04A9  1472          .ADDRESS -                      ; Address of FAO argument.
             000004D7'   04A9  1473                  TPARSE_BLK+TPA$L_TOKENCNT
                         04AD  1474
                         04AD  1475 MSG_OUT_ADRS:
             00000000    04AD  1476          .LONG   0                       ; Address to receive OUTADRS
                         04B1  1477                                          ;  information from $GETMSG.
                         04B1  1478
                         04B1  1479 ;
                         04B1  1480 ; Broadcast messages.
                         04B1  1481 ;
                         04B1  1482
             00000200    04B1  1483 MAX_MSG:.LONG    512                     ; Max message length.
                         04B5  1484
             00000200    04B5  1485 BUF_QUO:.LONG    512                     ; Number of system memory bytes that
                         04B9  1486                                          ;  can be used to buffer messages.
                         04B9  1487
                         04B9  1488 ;
                         04B9  1489 ; Parsing
```

```
                                 04B9  1490 ;
                                 04B9  1491
                                 04B9  1492 CMD_STATUS:                          ; Status returned from LIB$TPARSE.
                        00000000 04B9  1493         .LONG   0                   ;
                                 04BD  1494
                                 04BD  1495 TYPEAHD_BUF:                         ; Buffer to receive type ahead info
                        00000000 00000000 04BD  1496         .QUAD   0          ;  from sensemode.
                                 04C5  1497
                                 04C5  1498 INSTRNG_LEN:                         ; Word to receive length of input string
                            0000 04C5  1499         .WORD   0                   ;
                                 04C7  1500
                                 04C7  1501 TPARSE_BLK:                          ; TPARSE parameter block.
                        00000008 04C7  1502         .LONG   TPA$K_COUNTO        ; Longword count.
                        00000002 04CB  1503         .LONG   TPA$M_ABBREV        ; Allow abbreviation.
                        0000053F 04CF  1504         .BLKL   TPA$K_LENGTH0-8     ; Reserve room for parameters.
                                 053F  1505
                        00000000 053F  1506 KEY_VAL:.LONG   0                   ; Longword to receive keyword value
                                 0543  1507                                     ;  from LIB$LOOKUP_KEY.
                                 0543  1508
                        00000000 0543  1509 KEY_OFF:.LONG   0                   ; Address of longword containing the bit
                                 0547  1510                                     ;  which corresponds to the keyword.
                                 0547  1511
                        00000000 0547  1512 KEY_MSK:.LONG   0                   ; Mask of the bit corresponding to the
                                 054B  1513                                     ;  keyword.
                                 054B  1514
                                 054B  1515 KEY_DSCR:                            ; Address of the field descriptor
                        00000000 054B  1516         .LONG   0                   ;  corresponding to the keyword.
                                 054F  1517
                        00000000 054F  1518 VALUE:  .LONG   0                   ; Longword to receive value for SET
                                 0553  1519                                     ;  FIELD, SET INTERVAL, or SET SCREEN
                                 0553  1520                                     ;  command.
                                 0553  1521
                                 0553  1522 HELP_DSCR:                           ; Descriptor for help keywords.
                        00000000 00000000 0553  1523         .QUAD   0          ;
                                 055B  1524
                                 055B  1525 ;
                                 055B  1526 ; Descriptors for display buffers.
                                 055B  1527 ;
                                 055B  1528
                                 055B  1529 SCREEN_DSCR:                         ; Descriptor for SCREEN_BUF - the
                        00000200' 055B  1530         .LONG   SCREEN_BUF_LEN     ;  buffer for the screen package.
                                 055F  1531         .ADDRESS -                 ;
                        00000563' 055F  1532                 SCREEN_BUF         ;
                                 0563  1533 SCREEN_BUF:                          ;
                        00000763 0563  1534         .BLKB   512                ;
                        00000200 0763  1535 SCREEN_BUF_LEN = .-SCREEN_BUF       ;
                                 0763  1536
                                 0763  1537 CLASS_BOX_DSCR:                      ; Descriptor for CLASS_BOX_BUF - the
                        00000200' 0763  1538         .LONG   CLASS_BOX_LEN      ;  buffer for line above the classes in
                                 0767  1539         .ADDRESS -                 ;  the header.
                        00000768' 0767  1540                 CLASS_BOX_BUF      ;
                                 076B  1541 CLASS_BOX_BUF:                       ;
                        00000968 076B  1542         .BLKB   512                ;
                        00000200 096B  1543 CLASS_BOX_LEN = .-CLASS_BOX_BUF     ;
                                 096B  1544
                                 096B  1545 FIELD_TOP_BOX_DSCR:                  ; Descriptor for FIELD_TOP_BOX_BUF - the
                        00000200' 096B  1546         .LONG   FIELD_TOP_BOX_LEN  ;  buffer for the line above the field
```

```
                     096F  1547          .ADDRESS -                              ;  titles in the header.
00000973'            096F  1548                   FIELD_TOP_BOX_BUF              ;
                     0973  1549 FIELD_TOP_BOX_BUF:                               ;
00000B73             0973  1550          .BLKB   512                             ;
00000200             0B73  1551 FIELD_TOP_BOX_LEN = .-FIELD_TOP_BOX_BUF          ;
                     0B73  1552                                                  ;
                     0B73  1553 FIELD_BOT_BOX_DSCR:                              ; Descriptor for FIELD_BOT_BOX_BUF - the
00000200'            0B73  1554          .LONG   FIELD_BOT_BOX_LEN               ;  buffer for the line below the field
                     0B77  1555          .ADDRESS -                              ;  title in the header.
00000B7B'            0B77  1556                   FIELD_BOT_BOX_BUF              ;
                     0B7B  1557 FIELD_BOT_BOX_BUF:                               ;
00000D7B             0B7B  1558          .BLKB   512                             ;
00000200             0D7B  1559 FIELD_BOT_BOX_LEN = .-FIELD_BOT_BOX_BUF          ;
                     0D7B  1560                                                  ;
                     0D7B  1561 LAST_LINE_DSCR:                                  ; Descriptor for LAST_LINE_BUF - the
00000200'            0D7B  1562          .LONG   LAST_LINE_LEN                   ;  buffer for the line following the
                     0D7F  1563          .ADDRESS -                              ;  last line of data.
00000D83'            0D7F  1564                   LAST_LINE_BUF                  ;
                     0D83  1565 LAST_LINE_BUF:                                   ;
0C000F83             0D83  1566          .BLKB   512                             ;
00000200             0F83  1567 LAST_LINE_LEN = .-LAST_LINE_BUF                  ;
                     0F83  1568                                                  ;
                     0F83  1569 LAST_CLUB_LINE_DSCR:                             ; Descriptor for LAST_CLUB_LINE_BUF -
00000200'            0F83  1570          .LONG   LAST_CLUB_LINE_LEN              ;  the buffer for the line following the
                     0F87  1571          .ADDRESS -                              ;  last line of data in the CLUB
00000F8B'            0F87  1572                   LAST_CLUB_LINE_BUF             ;  portion of the CLUSTER report.
                     0F8B  1573 LAST_CLUB_LINE_BUF:                              ;
0000118B             0F8B  1574          .BLKB   512                             ;
00000200             118B  1575 LAST_CLUB_LINE_LEN =    .-LAST_CLUB_LINE_BUF
                     118B  1576                                                  ;
                     118B  1577 TITLE_DSCR:                                      ; Descriptor for TITLE_BUF - the buffer
00000400'            118B  1578          .LONG   TITLE_BUF_LEN                   ;  which receives the titles for
                     118F  1579          .ADDRESS -                              ;  display.
00001193'            118F  1580                   TITLE_BUF                      ;
                     1193  1581 TITLE_BUF:                                       ;
00001593             1193  1582          .BLKB   1024                            ;
00000400             1593  1583 TITLE_BUF_LEN = .-TITLE_BUF                      ;
                     1593  1584                                                  ;
                     1593  1585 CLASS_DSCR:                                     ; Descriptor for CLASS_BUF - the buffer
00000400'            1593  1586          .LONG   CLASS_BUF_LEN                   ;  which receives the class names for
                     1597  1587          .ADDRESS -                              ;  display.
0000159B'            1597  1588                   CLASS_BUF                      ;
                     159B  1589 CLASS_BUF:                                       ;
0000199B             159B  1590          .BLKB   1024                            ;
00000400             199B  1591 CLASS_BUF_LEN = .-CLASS_BUF                      ;
                     199B  1592                                                  ;
                     199B  1593 DATA_DSCR:                                       ; Descriptor for DATA_BUF the buffer
00000400'            199B  1594          .LONG   DATA_BUF_LEN                    ;  which receives the data for display.
                     199F  1595          .ADDRESS -                              ;
000019A3'            199F  1596                   DATA_BUF                       ;
                     19A3  1597 DATA_BUF:                                        ;
00001DA3             19A3  1598          .BLKB   1024                            ;
00000400             1DA3  1599 DATA_BUF_LEN = .-DATA_BUF                        ;
                     1DA3  1600                                                  ;
                     1DA3  1601
                     1DA3  1602 ;
                     1DA3  1603 ; Miscellaneous descriptors.
```

```
                1DA3  1604 ;
                1DA3  1605
                1DA3  1606 GET_VAL_DSCR:                        ; Descriptor for call to CLI$GET_VALUE.
0000003F'       1DA3  1607          .LONG   RET_BUF_LEN         ;
                1DA7  1608          .ADDRESS -                  ;
00001DB3'       1DA7  1609                  RET_BUF             ;
                1DAB  1610
                1DAB  1611 RET_BUF_DSCR:                        ; Descriptor for buffer to receive
0000003F'       1DAB  1612          .LONG   RET_BUF_LEN         ;  string from CLI$GET_VALUE.
                1DAF  1613          .ADDRESS -                  ;
00001DB3'       1DAF  1614                  RET_BUF             ;
00001DF2        1DB3  1615 RET_BUF:.BLKB    63                  ;
0000003F        1DF2  1616 RET_BUF_LEN = .-RET_BUF              ;
                1DF2  1617
                1DF2  1618 LAST_BRDCST_DSCR:                    ; Descriptor for a copy of the last
00000000        1DF2  1619          .LONG   0                   ;  broadcast message to be used to
                1DF6  1620          .ADDRESS -                  ; scroll the broadcast messages on
00001DFA'       1DF6  1621                  LAST_BRDCST_BUF     ; a VT52.
                1DFA  1622 LAST_BRDCST_BUF:                     ;
00001E7E        1DFA  1623          .BLKB   132                 ;
                1E7E  1624
                1E7E  1625
                1E7E  1626 ERR_MSG_DSCR:                        ; Descriptor used to output an error
00000000        1E7E  1627          .LONG   0                   ;  message.
                1E82  1628          .ADDRESS -                  ;
00001E8E'       1E82  1629                  GET_MSG_BUF         ;
                1E86  1630
                1E86  1631
                1E86  1632 GET_MSG_DSCR:                        ; Descriptor for buffer for $GETMSG.
00000100'       1E86  1633          .LONG   GET_MSG_BUF_LEN     ;
                1E8A  1634          .ADDRESS -                  ;
00001E8E'       1E8A  1635                  GET_MSG_BUF         ;
                1E8E  1636 GET_MSG_BUF:                         ;
00001F8E        1E8E  1637          .BLKB   256                 ;
00000100        1F8E  1638 GET_MSG_BUF_LEN = .-GET_MSG_BUF      ;
                1F8E  1639
                1F8E  1640
                1F8E  1641 ASC_TIM_DSCR:                        ; Descriptor used to format INCN_TIME.
00000017        1F8E  1642          .LONG   23                  ; Enough room for DD-MMM-YYYY HH:MM.
                1F92  1643          .ADDRESS -                  ;
00001F96'       1F92  1644                  TIMBUF              ;
00001FAD        1F96  1645 TIMBUF: .BLKB    23                  ;
                1FAD  1646
                1FAD  1647 MBX_MSG_DSCR:                        ; Descriptor for mailbox message buffer.
00000000        1FAD  1648          .LONG   0                   ; Reserved to receive message length.
00000000        1FB1  1649          .LONG   0                   ; Address of broadcast message (filled
                1FB5  1650                                      ;  in by OUT_MSG routine).
                1FB5  1651
00002085        1FB5  1652 MBX_BUF:.BLKB    256                 ; Buffer for receiving mailbox messages.
00000100        20B5  1653 MBX_BUF_LEN = .-MBX_BUF              ;
                20B5  1654
                20B5  1655 TIME_DSCR:                           ; Descriptor for buffer to receive
00000014'       20B5  1656          .LONG   TIME_LEN            ;  current date and time from
                20B9  1657          .ADDRESS -                  ;  LIB$DATE_TIME.
000020BD'       20B9  1658                          TIME_BUF    ;
000020D1        20BD  1659 TIME_BUF:.BLKB   20                  ;
00000014        20D1  1660 TIME_LEN = .-TIME_BUF                ;
```

```
                     20D1  1661
                     20D1  1662 DDM_SIZE:                               ; Size of the display discription
                0000 20D1  1663         .WORD   0                       ;  message.
                     20D3  1664
                     20D3  1665 CMD_LINE_DSCR:                          ; Descriptor for command input buffer.
          00000078' 20D3  1666         .LONG   CMD_BUF_LEN             ;
                     20D7  1667         .ADDRESS -                      ;
          000020DB' 20D7  1668                 CMD_BUF                 ;
                     20DB  1669 CMD_BUF:                                ;
          00002153  20DB  1670         .BLKB   120                     ;
          00000078  2153  1671 CMD_BUF_LEN = .-CMD_BUF                 ;
                     2153  1672
                     2153  1673 FORMAT_BUF_DSCR:                       ; Descriptor for FORMAT_BUF - used to
                     2153  1674 FORMAT_BUF_LEN:                        ;  store the data after formatting the
          00000000  2153  1675         .LONG   0                       ;  data in ascii.
                     2157  1676         .ADDRESS -                      ;
          0000215B' 2157  1677                 FORMAT_BUF              ;
                     215B  1678 FORMAT_BUF:                             ;
          000021AB  215B  1679         .BLKB   80                      ;
                     21AB  1680
                     21AB  1681
                     21AB  1682 ;
                     21AB  1683 ; Miscellaneous parameters.
                     21AB  1684 ;
                     21AB  1685
                00   21AB  1686 SWITCH: .BYTE   0                       ; Controls switching between reports
                     21AC  1687                                         ;  when /REPORT=CYCLE.  Reports are
                     21AC  1688                                         ;  switched when bit 0 is cleared.
                     21AC  1689
FFFFFFFF F70F2E80    21AC  1690 DELAY:  .LONG   -10*1000*1000*15,-1      ; Delay time between display updates
                     21B4  1691                                         ;  Defaults to fifteen seconds.
                     21B4  1692
00000000 00000000    21B4  1693 A_TIME: .QUAD   0                        ; Quadword to receive binary time for
                     21BC  1694                                         ;  /ENDING and /BEGINNING.
                     21BC  1695
                     21BC  1696 TITLE_COUNT:                            ; Used by GET_TITLES to calculate the
          00000000   21BC  1697         .LONG   0                       ;  length of the titles in characters.
                     21C0  1698
                     21C0  1699 TITLE_START:                            ; Used by GET_TITLES to calculate the
          00000000   21C0  1700         .LONG   0                       ;  length of the titles in bytes.
                     21C4  1701
                     21C4  1702 HEADER_SIZE:                            ; Size of the header in characters.
          00000000   21C4  1703         .LONG   0                       ;
                     21C8  1704
                     21C8  1705 CLUB_HEADER_SIZE:                       ; Size of CLUSTER class header.
          00000000   21C8  1706         .LONG   0                       ;
                     21CC  1707
                     21CC  1708 SCS_HEADER_SIZE:                        ; Size of SCS class header.
          00000000   21CC  1709         .LONG   0                       ;
                     21D0  1710 ;
                     21D0  1711 ; Array for storing the length in characters of the titles as calculated by
                     21D0  1712 ; the GET_TITLES and SAVE_TITLE_LEN routines.  This length is used by the
                     21D0  1713 ; GET_CLASSES routines to center the class name.
                     21D0  1714 ;
                     21D0  1715 ;                +--------+--------+--------+--------+
                     21D0  1716 ;                !       Size of CLUSTER titles      !
                     21D0  1717 ;                +--------+--------+--------+--------+
```

```
                     21D0  1718 ;                 ¦       Size of SYSTEMS titles      ¦
                     21D0  1719 ;                 +--------+-------+--------+--------+
                     21D0  1720 ;                 ¦       Size of MEMBERS titles      ¦
                     21D0  1721 ;                 +--------+-------+--------+--------+
                     21D0  1722 ;                 ¦      Size of CIRCUITS titles      ¦
                     21D0  1723 ;                 +--------+-------+--------+--------+
                     21D0  1724 ;                 ¦   Size of CONNECTIONS titles      ¦
                     21D0  1725 ;                 +--------+-------+--------+--------+
                     21D0  1726 ;                 ¦       Size of COUNTERS titles     ¦
                     21D0  1727 ;                 +--------+-------+--------+--------+
                     21D0  1728 ;                 ¦       Size of CREDITS titles      ¦
                     21D0  1729 ;                 +--------+-------+--------+--------+
                     21D0  1730 ;                 ¦    Size of LOCAL_PORTS titles     ¦
                     21D0  1731 ;                 +--------+-------+--------+--------+
                     21D0  1732 ;                 ¦       Size of ERRORS titles       ¦
                     21D0  1733 ;                 +--------+-------+--------+--------+
                     21D0  1734
                     21D0  1735 TITLE_LEN_ARRAY:
000021F4             21D0  1736          .BLKL   9
                     21F4  1737
                     21F4  1738 ;
                     21F4  1739 ; Miscellaneous parameters for screen display.
                     21F4  1740 ;
                     21F4  1741
                     21F4  1742 CURRENT_CLASS:                       ; Indicates which class is currently
00000000             21F4  1743          .LONG   0                   ;  being processed for display.
                     21F8  1744
                     21F8  1745 DATA_CNTRL:                          ; Controls which classes will be
00000000             21F8  1746          .LONG   0                   ;  processed for data display.
                     21FC  1747
                     21FC  1748 SAVED_CLASSES:                       ; Used to save class selections when
00000000             21FC  1749          .LONG   0                   ;  generating the display.
                     2200  1750
                     2200  1751 LOCAL_SAVE:                         ; Used to save local class selections
00000000             2200  1752          .LONG   0                   ;  when switching to cluster report.
                     2204  1753
                     2204  1754 CLUST_SAVE:                         ; Used to save cluster report
00000000             2204  1755          .LONG   0                   ;  selections when switching to the
                     2208  1756                                      ;  local report.
                     2208  1757
                     2208  1758
                     2208  1759 ;
                     2208  1760 ; Buffer for device information from $GETDVI.
                     2208  1761 ;
                     2208  1762
                     2208  1763 DEVICE_CLASS:                       ; Device class returned from $GETDVI.
00000000             2208  1764          .LONG   0                   ;
                     220C  1765 DEVICE_TYPE:                         ; Device type returned from $GETDVI.
00000000             220C  1766          .LONG   0                   ;
                     2210  1767 BUFFER_SIZE:                        ; Width of the terminals screen
00000000             2210  1768          .LONG   0                   ;  returned from $GETDVI.
                     2214  1769
                     2214  1770 SAVE_CHAR_BUF:
          00         2214  1771          .BYTE   0                   ; Device class.
          00         2215  1772          .BYTE   0                   ; Device type.
        0000         2216  1773          .WORD   0                   ; Screen width.
                     2218  1774 DEVDEPEND:                          ; Device terminal characteristics
```

```
00000000  2218  1775          .LONG   0                       ;  returned from $GETDVI.
          221C  1776 DEVDEPEND2:                            ; Extended terminal characteristics
00000000  221C  1777          .LONG   0                       ;  returned from $GETDVI.
0000000C  2220  1778 CHAR_BUF_LEN = .-SAVE_CHAR_BUF
          2220  1779
          2220  1780
          2220  1781 NEW_CHAR_BUF:                           ; Reserve space for a copy of
00002222  2220  1782          .BLKB   2                       ; Device class, type.
          2222  1783 SCREEN_WIDTH:                          ;
    0000  2222  1784          .WORD   0                       ; Screen width.
0000222C  2224  1785          .BLKL   2                       ; Characteristics, extended char.
          222C  1786
          222C  1787 OUTPUT_WIDTH:                          ; Output width for LBR$OUTPUT_HELP.
00000000  222C  1788          .LONG   0                       ;
          2230  1789
          2230  1790
          2230  1791 ;
          2230  1792 ; Buffer for process information from $GETJPI.
          2230  1793 ;
          2230  1794
          2230  1795 CUR_PRIV:                              ; Mask of processes current privileges.
00000000  2230  1796          .LONG   0                       ;
          2234  1797
          2234  1798 ;
          2234  1799 ; Storage area for various terminal parameters.
          2234  1800 ;
          2234  1801
    0000  2234  1802 IN_CHAN:.WORD    0                       ; Word to receive channel number for
          2236  1803                                        ;  terminal input.
          2236  1804
          2236  1805 OUT_CHAN:                              ; Word to receive channel number for
    0000  2236  1806          .WORD   0                       ;  terminal output.
          2238  1807
          2238  1808 SET_CHAN:                              ; Required by LIB$SET_OUTPUT.
00000000  2238  1809          .LONG   0                       ;
          223C  1810
    0000  223C  1811 MBXCHAN:.WORD    0                       ; Word to receive channel number
          223E  1812                                        ;  assigned to the mailbox.
          223E  1813
00000000 00000000 223E 1814 IOSB:     .QUAD   0             ; Reserve one quadword for I/O status
          2246  1815                                        ;  block.
          2246  1816
      00  2246  1817 IN_DEV: .BYTE    0                     ; Input device type.
      00  2247  1818 OUT_DEV:.BYTE    0                     ; Output device type.
          2248  1819
          2248  1820 LINE_NUM:                              ; Current line number
00000001  2248  1821          .LONG   1                       ;
          224C  1822 COL_NUM:                               ; Current column number
00000001  224C  1823          .LONG   1                       ;
          2250  1824 FIRST_LINE:                            ; First line on the screen.
00000001  2250  1825          .LONG   1                       ;
          2254  1826 FIRST_DATA_LINE:                       ; Number of the first line to contain
00000007  2254  1827          .LONG   7                       ; display data which can be vertically
          2258  1828                                        ;  scrolled.
          2258  1829 LAST_DATA_LINE:                        ; Number of the last line used to
00000000  2258  1830          .LONG   0                       ;  display data. (3rd to last on line
          225C  1831                                        ;  on the terminal)
```

```
                225C  1832 CMD_LINE:                                ; Number of the second to last line on
00000000        225C  1833          .LONG   0                       ;  the terminal.
                2260  1834 LAST_LINE:                               ; Number of the last line on the
00000000        2260  1835          .LONG   0                       ;  terminal.
                2264  1836
                2264  1837
                2264  1838 ;
                2264  1839 ; Parameters used by the FIND_ENTRY and LOOKUP_FIELD routines.  These
                2264  1840 ; parameters control which field descriptors to use to display the report.
                2264  1841 ;
                2264  1842
                2264  1843 ADRS_TABLE:                              ; Address of table of field descriptor
00000000        2264  1844          .LONG   0                       ;  addresses.
                2268  1845 BITMAP_ADRS:                            ; Address of bitmap to be scanned.
00000000        2268  1846          .LONG   0
                226C  1847 MAP_LEN:                                 ; Length of the bitmap to be scanned.
00000000        226C  1848          .LONG   0
                2270  1849 START_BIT:                               ; Number of the bit to start the scan.
00000000        2270  1850          .LONG   0
                2274  1851 KEYTAB_ADRS:                            ; Address of table of keywords used to
00000000        2274  1852          .LONG   0                       ;  identify a field
                2278  1853
                2278  1854
                2278  1855 ;
                2278  1856 ; List of systems that have been removed from tne display by system ID number.
                2278  1857 ;
                2278  1858
                2278  1859 SYS_ID_LIST:
000022D8        2278  1860          .BLKB   6*MAX_ID                ; Reserve six bytes for each possible
00000060        22D8  1861 ID_LIST_LEN = .-SYS_ID_LIST
                22D8  1862                                          ;  system.
00000010        22D8  1863 ID_LIST_ENTRYS  =       MAX_ID           ; Maximum number of entries in the list.
                22D8  1864
                22D8  1865 ID_COUNT:                                ; Number of entries in the list.
            00000000  22D8  1866          .LONG   0
                22DC  1867
                22DC  1868 ID_RESULT:                               ; Reserve one quadword for result from
00000000 00000000       22DC  1869          .QUAD   0               ;  system ID string to hex conversion.
                        22E4  1870
                        22E4  1871
```

```
22E4  1873          .SBTTL  Macro definitions.
22E4  1874          .SBTTL              -       Conditional Branch Macro.
22E4  1875
22E4  1876 ;
22E4  1877 ; The following macros are used in place of a BRANCH ON (CONDITION) instruction
22E4  1878 ; when the displacement exceeds 256 bytes.
22E4  1879 ;
22E4  1880
22E4  1881 ;
22E4  1882 ; JLBS - used in place of BLBS
22E4  1883 ;
22E4  1884
22E4  1885          .MACRO  JLBS    SOURCE,ADDRESS?L1
22E4  1886                  BLBC    SOURCE,L1
22E4  1887                  BRW     ADDRESS
22E4  1888 L1:
22E4  1889          .ENDM   JLBS
22E4  1890
22E4  1891
22E4  1892 ;
22E4  1893 ; JLBC - used in place of BLBC
22E4  1894 ;
22E4  1895
22E4  1896          .MACRO  JLBC    SOURCE,ADDRESS?L1
22E4  1897                  BLBS    SOURCE,L1
22E4  1898                  BRW     ADDRESS
22E4  1899 L1:
22E4  1900          .ENDM   JLBC
22E4  1901
22E4  1902
22E4  1903 ;
22E4  1904 ; JNEQ - used in place of BNEQ
22E4  1905 ;
22E4  1906
22E4  1907          .MACRO  JNEQ    ADDRESS?L1
22E4  1908                  BEQL    L1
22E4  1909                  BRW     ADDRESS
22E4  1910 L1:
22E4  1911          .ENDM   JNEQ
22E4  1912
22E4  1913
22E4  1914 ;
22E4  1915 ; JEQL - used in place of BEQL
22E4  1916 ;
22E4  1917
22E4  1918          .MACRO  JEQL    ADDRESS?L1
22E4  1919                  BNEQ    L1
22E4  1920                  BRW     ADDRESS
22E4  1921 L1:
22E4  1922          .ENDM   JEQL
22E4  1923
22E4  1924
22E4  1925 ;
22E4  1926 ; JNEQU - used in place of BNEQU
22E4  1927 ;
22E4  1928
22E4  1929          .MACRO  JNEQU   ADDRESS?L1
```

```
22E4  1930                      BEQLU   L1
22E4  1931                      BRW     ADDRESS
22E4  1932 L1:
22E4  1933              .ENDM   JNEQU
22E4  1934
22E4  1935
22E4  1936 ;
22E4  1937 ; JEQLU - used in place of BEQLU
22E4  1938 ;
22E4  1939
22E4  1940              .MACRO  JEQLU   ADDRESS?L1
22E4  1941                      BNEQU   L1
22E4  1942                      BRW     ADDRESS
22E4  1943 L1:
22E4  1944              .ENDM   JEQLU
22E4  1945
22E4  1946
22E4  1947 ;
22E4  1948 ; JGTR - used in place of BGTR
22E4  1949 ;
22E4  1950
22E4  1951              .MACRO  JGTR    ADDRESS?L1
22E4  1952                      BLEQ    L1
22E4  1953                      BRW     ADDRESS
22E4  1954 L1:
22E4  1955              .ENDM   JGTR
22E4  1956
22E4  1957
22E4  1958 ;
22E4  1959 ; JLEQ - Used In place of BLEQ
22E4  1960 ;
22E4  1961
22E4  1962              .MACRO  JLEQ    ADDRESS?L1
22E4  1963                      BGTR    L1
22E4  1964                      BRW     ADDRESS
22E4  1965 L1:
22E4  1966              .ENDM   JLEQ
22E4  1967
22E4  1968
22E4  1969 ;
22E4  1970 ; JGTRU - used in place of BGTRU
22E4  1971 ;
22E4  1972
22E4  1973              .MACRO  JGTRU   ADDRESS?L1
22E4  1974                      BLEQU   L1
22E4  1975                      BRW     ADDRESS
22E4  1976 L1:
22E4  1977              .ENDM   JGTRU
22E4  1978
22E4  1979
22E4  1980 ;
22E4  1981 ; JLEQU - used in place of BLEQU
22E4  1982 ;
22E4  1983
22E4  1984              .MACRO  JLEQU   ADDRESS?L1
22E4  1985                      BGTRU   L1
22E4  1986                      BRW     ADDRESS
```

```
22E4  1987 L1:
22E4  1988          .ENDM   JLEQU
22E4  1989
22E4  1990
22E4  1991 ;
22E4  1992 ; JGEQ - used in place of BGEQ
22E4  1993 ;
22E4  1994
22E4  1995          .MACRO  JGEQ    ADDRESS?L1
22E4  1996                  BLSS    L1
22E4  1997                  BRW     ADDRESS
22E4  1998 L1:
22E4  1999          .ENDM   JGEQ
22E4  2000
22E4  2001
22E4  2002 ;
22E4  2003 ; JLSS - used in place of BLSS
22E4  2004 ;
22E4  2005
22E4  2006          .MACRO  JLSS    ADDRESS?L1
22E4  2007                  BGEQ    L1
22E4  2008                  BRW     ADDRESS
22E4  2009 L1:
22E4  2010          .ENDM   JLSS
22E4  2011
22E4  2012
22E4  2013 ;
22E4  2014 ; JGEQU - used in place of BGEQU
22E4  2015 ;
22E4  2016
22E4  2017          .MACRO  JGEQU   ADDRESS?L1
22E4  2018                  BLSSU   L1
22E4  2019                  BRW     ADDRESS
22E4  2020 L1:
22E4  2021          .ENDM   JGEQU
22E4  2022
22E4  2023
22E4  2024 ;
22E4  2025 ; JLSSU - used in place of BLSSU
22E4  2026 ;
22E4  2027
22E4  2028          .MACRO  JLSSU   ADDRESS?L1
22E4  2029                  BGEQU   L1
22E4  2030                  BRW     ADDRESS
22E4  2031 L1:
22E4  2032          .ENDM   JLSSU
22E4  2033
22E4  2034
22E4  2035 ;
22E4  2036 ; JCS - used in place of BCS
22E4  2037 ;
22E4  2038
22E4  2039          .MACRO  JCS     ADDRESS?L1
22E4  2040                  BCC     L1
22E4  2041                  BRW     ADDRESS
22E4  2042 L1:
22E4  2043          .ENDM   JCS
```

```
                     ;
                     ; JCC - used in place of BCC
                     ;

                     .MACRO  JCC     ADDRESS?L1
                             BCS     L1
                             BRW     ADDRESS
                     L1:
                     .ENDM   JCC


                     ;
                     ; JVS - used in place of BVS
                     ;

                     .MACRO  JVS     ADDRESS?L1
                             BVC     L1
                             BRW     ADDRESS
                     L1:
                     .ENDM   JVS


                     ;
                     ; JVC - used in place of BVC
                     ;

                     .MACRO  JVC     ADDRESS?L1
                             BVS     L1
                             BRW     ADDRESS
                     L1:
                     .ENDM   JVC

                     ;
                     ; JBC - used in place of BBC
                     ;

                     .MACRO  JBC     BIT_NUM,SRC_ADRS,ADDRESS?L1
                             BBS     BIT_NUM,SRC_ADRS,L1
                             BRW     ADDRESS
                     L1:
                     .ENDM   JBC

                     ;
                     ; JBS - used in place of BBS
                     ;

                     .MACRO  JBS     BIT_NUM,SRC_ADRS,ADDRESS?L1
                             BBC     BIT_NUM,SRC_ADRS,L1
                             BRW     ADDRESS
                     L1:
                     .ENDM   JBS


```

```
22E4  2100          .SBTTL          -       DEFINE_FIELD macro.
22E4  2101 ;
22E4  2102 ; MACRO DEFINE_FIELD
22E4  2103 ;
22E4  2104 ;       This macro builts the data structures neccessary to display each field.
22E4  2105 ;
22E4  2106 ;
22E4  2107 ; ARGUMENTS -
22E4  2108 ;
22E4  2109 ;       REPORT -
22E4  2110 ;               Report that the field will appear in.  Must be CLUSTER,
22E4  2111 ;               or LOCAL_PORTS.
22E4  2112 ;
22E4  2113 ;       OFFSET -
22E4  2114 ;               Name of the SCS offset to use when extracting the data from the
22E4  2115 ;               SCS database.  This can be any predefined offset.  It is
22E4  2116 ;               normally one of the SCS offsets with prefixes CLUB$, SB$, CSB$,
22E4  2117 ;               PB$, CDT$, or PDT$ or a miscellaneous buffer offset with prefix
22E4  2118 ;               MB$.  This allows the default display routine DISPLAY_FIELD to
22E4  2119 ;               be used.
22E4  2120 ;
22E4  2121 ;       LENGTH -
22E4  2122 ;               The number of bytes of data to be extracted from the SCS
22E4  2123 ;               database to display the field.
22E4  2124 ;
22E4  2125 ;               Note: For ASCIC formated fields, LENGTH must be the maximum
22E4  2126 ;               number of characters in the length of the formatted string.
22E4  2127 ;               This may differ from the number of bytes extracted if the
22E4  2128 ;               DISPLAY_FIELD routine is not used for an ASCIC field.
22E4  2129 ;
22E4  2130 ;       TITLE -
22E4  2131 ;               Title for this field as it will appear in the display.
22E4  2132 ;
22E4  2133 ;       FORMAT -
22E4  2134 ;               Format of display for this field.  Must be one of HEX, DEC,
22E4  2135 ;               ASCII, ASCIC, or TIME.
22E4  2136 ;
22E4  2137 ;       MINIMUM_SIZE -
22E4  2138 ;               Minimum number of columns to allow for display of this field.
22E4  2139 ;
22E4  2140 ;       DEFAULT_SIZE -
22E4  2141 ;               Default number of columns to use for display of this field.
22E4  2142 ;
22E4  2143 ;       DISPLAY_FLAG -
22E4  2144 ;               This is an optional argument.  It is non-zero if the display of
22E4  2145 ;               the field is to be enabled in the report by default.  If this
22E4  2146 ;               argument is omitted, the field will not be displayed.
22E4  2147 ;
22E4  2148 ;       DISPLAY_ROUTINE -
22E4  2149 ;               Optional address of routine to handle display of this field.
22E4  2150 ;               If this argument is omitted, the field is displayed using the
22E4  2151 ;               routine DISPLAY_FIELD.
22E4  2152 ;
22E4  2153 ;
22E4  2154 ;       Five structures are built for each report:
22E4  2155 ;
22E4  2156 ;       'REPORT'_DSCRS -
```

```
22E4  2157 ;                This structure contains the field descriptors which contain the
22E4  2158 ;                information needed to display each of the fields in the report.
22E4  2159 ;                The following illustrates the format of one such descriptor:
22E4  2160 ;
22E4  2161 ; 'OFFSET'_DSCR:
22E4  2162 ;                        +--------+--------+--------+--------+
22E4  2163 ;                        |  MAX   |  MAX   |  MIN   | FIELD_ |
22E4  2164 ;                        | FIELD_ | DATA_  | FIELD_ |  SIZE  |
22E4  2165 ;                        |  SIZE  |  SIZE  |  SIZE  |        |
22E4  2166 ;                        +--------+--------+--------+--------+
22E4  2167 ;                        |     RESERVED     | FORMAT | LENGTH |
22E4  2168 ;                        +--------+--------+--------+--------+
22E4  2169 ;                        |        FD$L_DISPLAY_ROUTINE        |
22E4  2170 ;                        +--------+--------+--------+--------+
22E4  2171 ;                        |           FD$L_CLASS               |
22E4  2172 ;                        +--------+--------+--------+--------+
22E4  2173 ;                        |           FD$L_OFFSET              |
22E4  2174 ;                        +--------+--------+--------+--------+
22E4  2175 ;                        |        FD$L_OFFSET_PRFX            |
22E4  2176 ;                        +--------+--------+--------+--------+
22E4  2177 ;
22E4  2178 ; 'TITLE'_ADRS:
22E4  2179 ;                        +--------+--------+--------+--------+
22E4  2180 ;                        |                           |  SIZE  |
22E4  2181 ;                        |                           +--------+
22E4  2182 ;                        ~           FD$T_TITLE                ~
22E4  2183 ;                        |                                     |
22E4  2184 ;                        |                                     |
22E4  2185 ;                        +--------+--------+--------+--------+
22E4  2186 ;
22E4  2187 ;
22E4  2188 ;       'REPORT'_TABLE -
22E4  2189 ;
22E4  2190 ;               Since the field descriptors are of variable length, this table
22E4  2191 ;               is used to locate the start of each descriptor.  It contains a
22E4  2192 ;               list of pointers to each of the field descriptors for the
22E4  2193 ;               report.
22E4  2194 ;
22E4  2195 ;       ^               +--------+--------+--------+--------+
22E4  2196 ;       |               |    ADDRESS OF FIELD DESCRIPTOR    |
22E4  2197 ;       |               +--------+--------+--------+--------+
22E4  2198 ; Number of fields                        .
22E4  2199 ;                                        .
22E4  2200 ;       |                                .
22E4  2201 ;       v                                .
22E4  2202 ;
22E4  2203 ;
22E4  2204 ;       'REPORT'_BITMAP -
22E4  2205 ;
22E4  2206 ;               This bitmap is used to control which field in the report are
22E4  2207 ;               displayed.  Each bit in the map corresponds to one of the
22E4  2208 ;               fields in the display.  Setting the bit turns the field on.
22E4  2209 ;
22E4  2210 ;       ^               +--------+--------+--------+--------+
22E4  2211 ;       |               |  ONE BIT FOR EACH FIELD IN REPORT |
22E4  2212 ;                       +--------+--------+--------+--------+
22E4  2213 ; Number of fields/32                    .
```

```
                        ;                                       .
                        ;                                       .
                        ;       v                               .
                        ;
                        ;
                        ;       'REPORT'_KEYTAB -
                        ;
                        ;               This is a table of keywords which is used to associate a field
                        ;               title with a bit in the bitmap.  The format of this table is
                        ;               compatable with LIB$LOOKUP_KEY.
                        ;
                        ;                       +--------+--------+--------+--------+
                        ;                       |            VECTOR_COUNT             |
                        ;                       +--------+--------+--------+--------+
                        ;       ^               +--------+--------+--------+--------+
                        ;       |               |            KEYWORD VALUE            |
                        ;       |               +--------+--------+--------+--------+
                        ; Number of fields       |   ADDRESS OF KEYWORD NAME STRING    |
                        ;   times two            +--------+--------+--------+--------+
                        ;                                       .
                        ;       |                               .
                        ;       v                               .
                        ;
                        ;       'REPORT'_SIZE_ARRAY -
                        ;
                        ;               This array is used to restore the default widths for each of the
                        ;               fields in the display.  It contains one byte for each of the
                        ;               fields in the display.  Each byte is a copy of the byte at
                        ;               FD$B_FIELD_SIZE in the field descriptor.
                        ;
                        ;       ^               +---------+
                        ;       |               | Copy of |
                        ;       |               | FIELD_  |
                        ;       |               |  SIZE   |
                        ; One byte for each     +---------+
                        ;     field                     .
                        ;                               .
                        ;       |                       .
                        ;       v
                        ;
                        ;
00000000                KEYWRDVAL       =       0

                                .NOSHOW MEB

                                .MACRO  DEFINE_FIELD    REPORT,OFFSET,LENGTH,TITLE,FORMAT-
                                                        MINIMUM_SIZE,DEFAULT_SIZE,DISPLAY_FLAG=0,-
                                                        DISPLAY_ROUTINE=DISPLAY_FIELD

                                .PSECT  'REPORT'_SIZE_ARRAY,RD,WRT,NOEXE

                                .BYTE   DEFAULT_SIZE                    ; Save the default field size for use
                                                                        ;  by the INITIALIZE command.

                                .PSECT  'REPORT'_DSCRS,RD,WRT,NOEXE
```

```
22E4  2271
22E4  2272 'OFFSET'_DSCR:
22E4  2273          .BYTE   DEFAULT_SIZE                ; Store the current display field size
22E4  2274                                              ;  at offset FD$B_FIELD_SIZE.
22E4  2275          .BYTE   MINIMUM_SIZE                ; Store the minimum display field size
22E4  2276                                              ; at offset FD$B_MIN_FLD_SIZE.
22E4  2277
22E4  2278 ;
22E4  2279 ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4  2280 ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using ASCII
22E4  2281 ; or hexadecimal format.
22E4  2282 ;
22E4  2283
22E4  2284          .IF     NE      FORMAT-DEC
22E4  2285          .IF     NE      FORMAT-ASCIC
22E4  2286          .IF     NE      FORMAT-TIME
22E4  2287          .BYTE   FORMAT*LENGTH
22E4  2288          .IF     GE      FORMAT*LENGTH-%LENGTH(TITLE)
22E4  2289          .BYTE   FORMAT*LENGTH
22E4  2290          .IFF
22E4  2291          .BYTE   %LENGTH(TITLE)
22E4  2292          .ENDC
22E4  2293          .ENDC
22E4  2294          .ENDC
22E4  2295          .ENDC
22E4  2296
22E4  2297 ;
22E4  2298 ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4  2299 ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using ASCIC
22E4  2300 ; format.
22E4  2301 ;
22E4  2302          .IF     EQ      FORMAT-ASCIC
22E4  2303          .BYTE   FORMAT/ASCIC*LENGTH
22E4  2304          .IF     GE      FORMAT/ASCIC*LENGTH-%LENGTH(TITLE)
22E4  2305          .BYTE   FORMAT/ASCIC*LENGTH
22E4  2306          .IFF
22E4  2307          .BYTE   %LENGTH(TITLE)
22E4  2308          .ENDC
22E4  2309          .ENDC
22E4  2310
22E4  2311 ;
22E4  2312 ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4  2313 ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed as a time
22E4  2314 ; string.
22E4  2315 ;
22E4  2316          .IF     EQ      FORMAT-TIME
22E4  2317          .BYTE   LENGTH
22E4  2318          .IF     GE      LENGTH-%LENGTH(TITLE)
22E4  2319          .BYTE   LENGTH
22E4  2320          .IFF
22E4  2321          .BYTE   %LENGTH(TITLE)
22E4  2322          .ENDC
22E4  2323          .ENDC
22E4  2324
22E4  2325 ;
22E4  2326 ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4  2327 ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using
```

```
22E4  2328 ; decimal format.
22E4  2329 ;
22E4  2330          .IF     EQ      FORMAT-DEC
22E4  2331         LEN = 1
22E4  2332          .IRP    SIZE,<3,5,8,10,13,15,17,20>
22E4  2333          .IIF    EQ      LENGTH-LEN,     DEC_SIZE        =       SIZE
22E4  2334         LEN = LEN +1
22E4  2335          .ENDR
22E4  2336          .BYTE   DEC_SIZE
22E4  2337          .IF     GE      DEC_SIZE-%LENGTH(TITLE)
22E4  2338          .BYTE   DEC_SIZE
22E4  2339          .ENDC
22E4  2340          .IF     LT      DEC_SIZE-%LENGTH(TITLE)
22E4  2341          .BYTE   %LENGTH(TITLE)
22E4  2342          .ENDC
22E4  2343          .ENDC
22E4  2344
22E4  2345          .BYTE   LENGTH                  ; Store the number of bytes that appear
22E4  2346                                          ;  in the SCS database for this field
22E4  2347                                          ;  at offset FD$B_LENGTH.
22E4  2348          .BYTE   FORMAT                  ; Store the type of format to use to
22E4  2349                                          ;  display the data at offset
22E4  2350                                          ;  FD$B_FORMAT.
22E4  2351          .WORD   0                       ; Reserve on one.
22E4  2352          .ADDRESS -                      ; Store the address of the routine used
22E4  2353                  DISPLAY_ROUTINE         ;  to display the field at offset
22E4  2354                                          ; FD$L_DISPLAY_ROUTINE.
22E4  2355          .LONG   CLASS                   ; Store the family type that this field
22E4  2356                                          ;  is a member of at offset FD$L_CLASS.
22E4  2357          .LONG   'OFFSET'                ; Store the value of the SCS data
22E4  2358                                          ;  structure offset for this field at
22E4  2359                                          ; offset FD$L_OFFSET.
22E4  2360 ;
22E4  2361 ; Save the first four characters of the offset name at offset
22E4  2362 ; FD$L_OFFSET_PRFX.
22E4  2363 ;
22E4  2364          .IIF EQ %LOCATE(<CLUB$>,OFFSET),        .LONG   ^A/CLUB/
22E4  2365          .IIF EQ %LOCATE(<SB$>,OFFSET),          .LONG   ^A/SB$ /
22E4  2366          .IIF EQ %LOCATE(<CSB$>,OFFSET),         .LONG   ^A/CSB$/
22E4  2367          .IIF EQ %LOCATE(<PB$>,OFFSET),          .LONG   ^A/PB$ /
22E4  2368          .IIF EQ %LOCATE(<CDT$>,OFFSET),         .LONG   ^A/CDT$/
22E4  2369          .IIF EQ %LOCATE(<PDT$>,OFFSET),         .LONG   ^A/PDT$/
22E4  2370          .IIF EQ %LOCATE(<MB$>,OFFSET),          .LONG   ^A/MB$ /
22E4  2371          .IIF EQ %LOCATE(<SCS$>,OFFSET),         .LONG   ^A/SCS$/
22E4  2372
22E4  2373          .IIF EQ %LOCATE(<$>,OFFSET) - %LENGTH(OFFSET),  .LONG   0
22E4  2374
22E4  2375 ;
22E4  2376 ; Store the ascii title at offset FD$T_TITLE.  Create a label for the
22E4  2377 ; location where the title is stored.  The label is the same as the title, but
22E4  2378 ; is followed by '_ADRS'.
22E4  2379 ;
22E4  2380
22E4  2381 'TITLE'_ADRS:
22E4  2382
22E4  2383
22E4  2384 ;
```

```
22E4  2385 ; Now save the title as an ASCIC string.  This will place the length of the
22E4  2386 ; string at offset FD$B_TITLE_SIZE and the string at offset FD$T_TITLE.
22E4  2387 ;
22E4  2388         .ASCIC  /TITLE/                 ; Title for display field.
22E4  2389                                         ; FD$B_TITLE_SIZE
22E4  2390                                         ; FD$T_TITLE
22E4  2391
22E4  2392         .PSECT  'REPORT'_TABLE,RD,NOWRT,NOEXE
22E4  2393
22E4  2394         .ADDRESS -                      ; Store descriptor address in the
22E4  2395                 'OFFSET'_DSCR           ;  address table.
22E4  2396 'REPORT'_TAB_LEN = .-'REPORT'_TABLE     ; Calculate length of the table.
22E4  2397
22E4  2398         .PSECT  'REPORT'_BITMAP,RD,WRT,NOEXE
22E4  2399
22E4  2400 ;
22E4  2401 ; Every 33 times we invoke this macro, save the mask we are building in the
22E4  2402 ; bitmap and reset the mask to 0.  Also increment the count of the number of
22E4  2403 ; longwords in the map.
22E4  2404 ;
22E4  2405
22E4  2406 COUNT             =       COUNT+1
22E4  2407         .IF      EQ      COUNT-33
22E4  2408         .LONG    MASK                   ; Save the mask in the bitmap.
22E4  2409
22E4  2410 MAP_LENGTH        =       MAP_LENGTH+1   ; Increase count of the number of
22E4  2411                                         ;  longwords in the map.
22E4  2412 MASK              =       0              ; Clear the mask.
22E4  2413 COUNT             =       1              ; Reinitialize the counter.
22E4  2414 MASK_BIT          =       1              ; Start over with bit 0.
22E4  2415
22E4  2416         .ENDC
22E4  2417
22E4  2418 ;
22E4  2419 ; If the display flag for this field descriptor is non-zero the this field is
22E4  2420 ; to be displayed.  Set the bit for this field by combining the bit value for
22E4  2421 ; this field with the bit mask.
22E4  2422 ;
22E4  2423
22E4  2424         .IF      NE      DISPLAY_FLAG
22E4  2425 MASK              =       MASK!MASK_BIT
22E4  2426         .ENDC
22E4  2427
22E4  2428 ;
22E4  2429 ; On to the next bit.
22E4  2430 ;
22E4  2431
22E4  2432 MASK_BIT          =       MASK_BIT@1
22E4  2433
22E4  2434
22E4  2435         .PSECT  'REPORT'_KEYTABLE
22E4  2436
22E4  2437
22E4  2438 ;
22E4  2439 ; Save the address of the title string in the keyword table.  This address is
22E4  2440 ; used by LIB%LOOKUP_KEY.
22E4  2441 ;
```

```
22E4  2442
22E4  2443          .ADDRESS -
22E4  2444                  'TITLE'_ADRS
22E4  2445
22E4  2446
22E4  2447 ;
22E4  2448 ; Save the keyword value in the keyword table.  This value is used by
22E4  2449 ; LIB$LOOKUP_KEY.
22E4  2450 ;
22E4  2451
22E4  2452          .LONG   KEYWRDVAL
22E4  2453
22E4  2454 KEYWRDVAL       =       KEYWRDVAL+1
22E4  2455
22E4  2456          .ENDM   DEFINE_FIELD
22E4  2457
22E4  2458
```

```
                     22E4  2460          .SBTTL  Cluster report field descriptors.
                     22E4  2461          .SAVE_PSECT
                     22E4  2462
                     22E4  2463 ;
                     22E4  2464 ; Initialize counters and masks.
                     22E4  2465 ;
                     22E4  2466
00000001             22E4  2467 MASK_BIT         =       1
00000000             22E4  2468 MASK             =       0
00000000             22E4  2469 COUNT            =       0
FFFFFFFF             22E4  2470 MAP_LENGTH       =       -1
                     22E4  2471
                     22E4  2472 ;
                     22E4  2473 ; Create labels which will be used to reference each of the data structures
                     22E4  2474 ; used in the cluster report as created by the DEFINE_FIELD macro.
                     22E4  2475 ;
                00000000  2476          .PSECT  CLUSTER_SIZE_ARRAY,RD,WRT,NOEXE
                    0000  2477 CLUSTER_SIZE_ARRAY:
                    0000  2478
                00000000  2479          .PSECT  CLUSTER_BITMAP,RD,WRT,NOEXE
                    0000  2480 CLUSTER_BITMAP:
                    0000  2481
                00000000  2482          .PSECT  CLUSTER_TABLE,RD,NOWRT,NOEXE
                    0000  2483 CLUSTER_TABLE:
                    0000  2484
                00000000  2485          .PSECT  CLUSTER_KEYTABLE,RD,NOWRT,NOEXE
                    0000  2486 CLUSTER_KEYTAB:
00000000            0000  2487          .LONG   0
                    0004  2488
                    0004  2489 ;
                    0004  2490 ; Define the fields that make up the CLUSTER class.
                    0004  2491 ;
                    0004  2492
00000001            0004  2493 CLASS    =       CLS$M_CLUB
                    0004  2494
                    0004  2495          DEFINE_FIELD    <CLUSTER>,<CLUB$W_QUORUM>,2,<CL_QUORUM>,DEC,4,4,1
                    000C  2496          DEFINE_FIELD    <CLUSTER>,<CLUB$W_VOTES>,2,<CL_VOTES>,DEC,4,4,1
                    0014  2497          DEFINE_FIELD    <CLUSTER>,<CLUB$T_QDNAME>,16,<QD_NAME>,ASCIC,6,15,1
                    001C  2498          DEFINE_FIELD    <CLUSTER>,<CLUB$L_FLAGS>,Y__,NO_STRNG_LEN,-
                    001C  2499                          <QF_VOTE>,ASCIC,4,4,1,DISPLAY_QF_VOTE
                    0024  2500          DEFINE_FIELD    <CLUSTER>,<CLUB$W_QDVOTES>,2,<CL_QDVOTES>,DEC,4,6
                    002C  2501          DEFINE_FIELD    <CLUSTER>,<CLUB$W_MEMSEQ>,2,<MEMSEQ>,DEC,4,6
                    0034  2502          DEFINE_FIELD    <CLUSTER>,<CLUB$Q_FTIME>,TIME_STRING_LEN,<FORMED>,-
                    0034  2503                          TIME,6,15,1
                    003C  2504          DEFINE_FIELD    <CLUSTER>,<CLUB$Q_LST_TIME>,TIME_STRING_LEN,-
                    003C  2505                          <LAST_TRANSITION>,TIME,6,15,1
                    0044  2506
                    0044  2507 ;
                    0044  2508 ; Define the fields that make up the SYSTEMS class.
                    0044  2509 ;
                    0044  2510
00000002            0044  2511 CLASS    =       CLS$M_SYS
                    0044  2512
                    0044  2513          DEFINE_FIELD    <CLUSTER>,<SB$T_NODENAME>,16,<NODE>,ASCIC,2,6,1
                    004C  2514          DEFINE_FIELD    <CLUSTER>,<SB$B_SYSTEMID>,6,<SYS_ID>,DEC,2,8
                    0054  2515          DEFINE_FIELD    <CLUSTER>,<SB$B_HWVERS>,12,<HW_VERS>,HEX,2,20
00000034            005C  2516 SB$T_HWTYPE = SB$Q_SWINCARN + 8
```

```
                        005C  2517          DEFINE_FIELD    <CLUSTER>,<SB$T_HWTYPE>,4,<HW_TYPE>,ASCII,4,4
                        0064  2518          DEFINE_FIELD    <CLUSTER>,<SB$T_SWVERS>,4,<SOFTWARE>,ASCII,3,8,1,-
                        0064  2519                          DISPLAY_SOFTWARE
                        006C  2520          DEFINE_FIELD    <CLUSTER>,<SB$SHC$L_NUM_PB>,2,<NUM_CIRCUITS>,DEC,7,7
                        0074  2521          DEFINE_FIELD    <CLUSTER>,<SB$W_MAXDG>,2,<DG_SIZE>,DEC,2,4
                        007C  2522          DEFINE_FIELD    <CLUSTER>,<SB$W_MAXMSG>,2,<MSG_SIZE>,DEC,2,5
                        0084  2523          DEFINE_FIELD    <CLUSTER>,<SB$Q_SWINCARN>,8,<INCARNATION>,HEX,2,14
0000002C                008C  2524 SB$Q_SWINCARN_2 = SB$Q_SWINCARN
                        008C  2525          DEFINE_FIELD    <CLUSTER>,<SB$Q_SWINCARN_2>,TIME_STRING_LEN,-
                        008C  2526                          <INCN_TIME>,TIME,6,9
                        0094  2527
                        0094  2528 ;
                        0094  2529 ; Define the fields that make up the MEMBERS class.
                        0094  2530 ;
                        0094  2531
00000004                0094  2532 CLASS           =       CLS$M_CSB
                        0094  2533
                        0094  2534          DEFINE_FIELD    <CLUSTER>,<CSB$L_CSID>,4,<CSID>,HEX,5,6
                        009C  2535          DEFINE_FIELD    <CLUSTER>,<CSB$W_SENDSEQNM>,2,<SEND_SQ>,DEC,6,6
                        00A4  2536          DEFINE_FIELD    <CLUSTER>,<CSB$W_RCVDSEQNM>,2,<RCVD_SQ>,DEC,6,6
                        00AC  2537          DEFINE_FIELD    <CLUSTER>,<CSB$W_ACKRSEQNM>,2,<ACKR_SQ>,DEC,4,6
                        00B4  2538          DEFINE_FIELD    <CLUSTER>,<CSB$W_LCKDIRWT>,2,<DIR_WT>,DEC,4,6
                        00BC  2539          DEFINE_FIELD    <CLUSTER>,<CSB$B_UNACKEDMSGS>,1,<UNACKED>,DEC,3,5
                        00C4  2540          DEFINE_FIELD    <CLUSTER>,<CSB$B_REMACKLIM>,1,<ACK_LM>,DEC,4,5
                        00CC  2541          DEFINE_FIELD    <CLUSTER>,<CSB$B_ECOLVL>,2,<PROTOCOL>,DEC,3,5,,-
                        00CC  2542                          DISPLAY_PROTOCOL
                        00D4  2543          DEFINE_FIELD    <CLUSTER>,<CSB$B_WARMCDRPS>,1,<WARMCDRPS>,DEC,1,5
                        00DC  2544          DEFINE_FIELD    <CLUSTER>,<CSB$B_STATE>,CSB_STATE_STRNG_LEN,-
                        00DC  2545                          <CNX_STATE>,ASCIC,5,6,,DISPLAY_CSB_STATE
                        00E4  2546          DEFINE_FIELD    <CLUSTER>,<CSB$W_VOTES>,2,<VOTES>,DEC,1,5
                        00EC  2547          DEFINE_FIELD    <CLUSTER>,<CSB$W_QUORUM>,2,<QUORUM>,DEC,1,6
                        00F4  2548          DEFINE_FIELD    <CLUSTER>,<CSB$W_QDVOTES>,2,<QDVOTES>,DEC,4,7
                        00FC  2549
00000060                00FC  2550 CSB$L_STATUS1   =       CSB$L_STATUS
                        00FC  2551          DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS1>,YES_NO_STRNG_LEN,-
                        00FC  2552                          <QF_SAME>,ASCIC,4,7,,DISPLAY_QF_SAME
00000060                0104  2553 CSB$L_STATUS2   =       CSB$L_STATUS
                        0104  2554          DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS2>,YES_NO_STRNG_LEN,-
                        0104  2555                          <QF_ACTIVE>,ASCIC,4,9,,DISPLAY_QF_ACTIVE
                        010C  2556          DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS>,CSB_STATUS_STRNG_LEN,-
                        010C  2557                          <STATUS>,ASCIC,3,7,1,DISPLAY_CSB_STATUS
                        0114  2558          DEFINE_FIELD    <CLUSTER>,<CSB$Q_REFTIME>,TIME_STRING_LEN,-
                        0114  2559                          <TRANSITION_TIME>,TIME,6,15,,
                        011C  2560 ;
                        011C  2561 ; Define the fields that make up the CIRCUITS class.
                        011C  2562 ;
                        011C  2563
00000008                011C  2564 CLASS           =       CLS$M_CIR
                        011C  2565
                        011C  2566          DEFINE_FIELD    <CLUSTER>,<PB$T_LPORT_NAME>,4,<LPORT_NAME>,ASCII,3,5
                        0124  2567          DEFINE_FIELD    <CLUSTER>,<PB$B_RSTATION>,6,<RPORT_NUM>,DEC,2,5,1
                        012C  2568          DEFINE_FIELD    <CLUSTER>,<PB$B_RST_PORT>,1,<RP_OWNER>,DEC,5,6
                        0134  2569          DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_TYP>,RPORT_TYP_STRNG_LEN,-
                        0134  2570                          <RP_TYPE>,ASCIC,5,6,1,DISPLAY_RPORT_TYP
                        013C  2571          DEFINE_FIELD    <CLUSTER>,<PB$SHC$L_NUM_CDT>,4,<NUM_CONNECTIONS>,DEC,7,7
                        0144  2572          DEFINE_FIELD    <CLUSTER>,<PB$W_STATE>,PB_STATE_STRNG_LEN,-
                        0144  2573                          <CIR_STATE>,ASCIC,5,7,1,DISPLAY_CIR_STATE
```

```
                     014C  2574          DEFINE_FIELD    <CLUSTER>,<PB$B_RSTATE>,PB_STATE_STRNG_LEN,-
                     014C  2575                          <REM_STATE>,ASCIC,6,7,,DISPLAY_RSTATE
                     0154  2576          DEFINE_FIELD    <CLUSTER>,<PB$B_CBL_STS>,CB_STAT_STRNG_LEN,-
                     0154  2577                          <CABLE_STATUS>,ASCIC,5,5,,DISPLAY_CABLE_STATUS
                     015C  2578          DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_REV>,4,<RP_REVISION>,HEX,5,8
                     0164  2579          DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_FCN>,4,<RP_FUNCTIONS>,HEX,5,8
                     016C  2580          DEFINE_FIELD    <CLUSTER>,<PB$L_WAITQFL>,4,<SCS_WAITERS>,DEC,2,5,,
                     0174  2581
                     0174  2582 ;
                     0174  2583 ; Define the fields that make up the CONNECTIONS class.
                     0174  2584 ;
                     0174  2585
00000010             0174  2586 CLASS            =          CLS$M_CON
                     0174  2587
                     0174  2588          DEFINE_FIELD    <CLUSTER>,<SCS$T_DST_PROC>,16,<LOC_PROC_NAME>,-
                     0174  2589                          ASCII,8,16,1,DISPLAY_PROC_NAME
                     017C  2590          DEFINE_FIELD    <CLUSTER>,<SCS$T_SRC_PROCS,16,<REM_PROC_NAME>,-
                     017C  2591                          ASCII,8,16,,DISPLAY_PROC_NAME
                     0184  2592          DEFINE_FIELD    <CLUSTER>,<CDT$W_STATE>,DISPLAY_CON_STATE_LEN,-
                     0184  2593                          <CON_STATE>,ASCIC,6,7,1,DISPLAY_CON_STATE
                     018C  2594          DEFINE_FIELD    <CLUSTER>,<CDT$L_LCONID>,4,<LOC_CONID>,HEX,8,9,
                     0194  2595          DEFINE_FIELD    <CLUSTER>,<CDT$L_RCONID>,4,<REM_CONID>,HEX,8,9,
                     019C  2596          DEFINE_FIELD    <CLUSTER>,<CDT$W_BLKSTATE>,DISPLAY_SCS_STATE_LEN,-
                     019C  2597                          <SCS_STATE>,ASCIC,6,7,,DISPLAY_SCS_STATE
                     01A4  2598
                     01A4  2599 ;
                     01A4  2600 ; Define the fields that make up the COUNTERS class.
                     01A4  2601 ;
                     01A4  2602
00000020             01A4  2603 CLASS            =          CLS$M_CNT
                     01A4  2604          DEFINE_FIELD    <CLUSTER>,<CDT$L_DGSENT>,4,<DGS_SENT>,DEC,5,5,1,
                     01AC  2605          DEFINE_FIELD    <CLUSTER>,<CDT$L_DGRCVD>,4,<DGS_RCVD>,DEC,5,5,1,
                     01B4  2606          DEFINE_FIELD    <CLUSTER>,<CDT$L_DGDISCARD>,4,<DGS_DSCRD>,DEC,-
                     01B4  2607                          5,5,,
                     01BC  2608          DEFINE_FIELD    <CLUSTER>,<CDT$L_MSGSENT>,4,<MSGS_SENT>,DEC,6,6,1,
                     01C4  2609          DEFINE_FIELD    <CLUSTER>,<CDT$L_MSGRCVD>,4,<MSGS_RCVD>,DEC,6,6,1,
                     01CC  2610          DEFINE_FIELD    <CLUSTER>,<CDT$L_SNDDATS>,4,<BLKS_SENT>,DEC,6,6,,
                     01D4  2611          DEFINE_FIELD    <CLUSTER>,<CDT$L_REQDATS>,4,<BLKS_REQ>,DEC,3,6,,
                     01DC  2612          DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTSENT>,4,<KB_SENT>,DEC,4,4,,-
                     01DC  2613                          DISPLAY_KBYTES
                     01E4  2614          DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTREQD>,4,<KB_RCVD>,DEC,4,4,,-
                     01E4  2615                          DISPLAY_KBYTES
                     01EC  2616          DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTMAPD>,4,<KB_MAPPED>,DEC,4,4,,-
                     01EC  2617                          DISPLAY_KBYTES
                     01F4  2618          DEFINE_FIELD    <CLUSTER>,<CDT$W_QCR_CNT>,2,<CR_WAITS>,DEC,1,4,,
                     01FC  2619          DEFINE_FIELD    <CLUSTER>,<CDT$W_QBDT_CNT>,2,<BDT_WAITS>,DEC,1,5,,
                     0204  2620
                     0204  2621 ;
                     0204  2622 ; Define the fields that make up the CREDITS class.
                     0204  2623 ;
                     0204  2624
00000040             0204  2625 CLASS            =          CLS$M_CRD
                     0204  2626
                     0204  2627          DEFINE_FIELD    <CLUSTER>,<CDT$W_SEND>,2,<SEND>,-
                     0204  2628                          DEC,1,4,1,
                     020C  2629          DEFINE_FIELD    <CLUSTER>,<CDT$W_MINSEND>,2,<MIN_SEND>,-
                     020C  2630                          DEC,5,5,1,
```

```
                0214  2631          DEFINE_FIELD    <CLUSTER>,<CDT$W_REC>,2,<RECEIVE>,-
                0214  2632                          DEC,1,3,1,
                021C  2633          DEFINE_FIELD    <CLUSTER>,<CDT$W_PENDREC>,2,<PEND_REC>,-
                021C  2634                          DEC,1,6,1,
                0224  2635          DEFINE_FIELD    <CLUSTER>,<CDT$W_INITLREC>,2,<INIT_REC>,-
                0224  2636                          DEC,1,6,1,
                022C  2637          DEFINE_FIELD    <CLUSTER>,<CDT$W_MINREC>,2,<MIN_REC>,-
                022C  2638                          DEC,5,5,1,
                0234  2639
      00000234  0234  2640          .PSECT  CLUSTER_KEYTABLE,RD,NOWRT,NOEXE
                0234  2641
00000046        0234  2642 NUM_CLUSTER_DSCRS = KEYWRDVAL             ; Save the number of descriptors in the
                0234  2643                                          ;  cluster report.
                0234  2644
00000000        0234  2645 . =       CLUSTER_KEYTAB                 ; Save the vector count for
0000008C        0000  2646          .LONG   KEYWRDVAL*2             ;  LIB$LOOKUP_KEY.
00000094        0004  2647 . =       .+KEYWRDVAL*2
                0094  2648
                0094  2649
                0094  2650 ;
                0094  2651 ; Save any partial mask in the bitmap and increase the count of the bitmap's
                0094  2652 ; length.
                0094  2653 ;
      00000008  0094  2654          .PSECT  CLUSTER_BITMAP,RD,WRT,NOEXE
                0098  2655
FFFFFFE5        0098  2656          .IF     NE      COUNT-33
0000003F        0098  2657          .LONG   MASK
00000002        009C  2658          MAP_LENGTH      =       MAP_LENGTH+1
                009C  2659          .ENDC
                009C  2660
00000002        009C  2661 CLUSTER_MAP_LEN = MAP_LENGTH             ; Save the number of longwords in the
                009C  2662                                          ;  cluster bitmap.
                009C  2663
                009C  2664 CL_SAVE_BITMAP:                          ; Reserve enough room for a copy of the
00000018        009C  2665          .BLKL   MAP_LENGTH+1            ;  cluster bitmap to be used by the
                00B8  2666                                          ;  INITIALIZE command.
                00B8  2667
                00B8  2668
```

```
                     0018  2670          .SBTTL  Local status report field descriptors.
                     0018  2671 ;
                     0018  2672 ; Initialize counters and masks.
                     0018  2673 ;
                     0018  2674
00000001             0018  2675 MASK_BIT        =       1
00000000             0018  2676 MASK            =       0
00000000             0018  2677 COUNT           =       0
FFFFFFFF             0018  2678 MAP_LENGTH      =       -1
00000000             0018  2679 KEYWRDVAL       =       0
                     0018  2680
                     0018  2681 ;
                     0018  2682 ; Create labels which will be used to reference each of the data structures
                     0018  2683 ; used in the local report as created by the DEFINE_FIELD macro.
                     0018  2684 ;
                     0018  2685
                00000000  2686          .PSECT  LOCAL_SIZE_ARRAY,RD,WRT,NOEXE
                     0000  2687 LOCAL_SIZE_ARRAY:
                     0000  2688
                00000000  2689          .PSECT  LOCAL_BITMAP,RD,WRT,NOEXE
                     0000  2690 LOCAL_BITMAP:
                     0000  2691
                00000000  2692          .PSECT  LOCAL_TABLE,RD,NOWRT,NOEXE
                     0000  2693 LOCAL_TABLE:
                     0000  2694
                00000000  2695          .PSECT  LOCAL_KEYTABLE,RD,NOWRT,NOEXE
                     0000  2696 LOCAL_KEYTAB:
00000000             0000  2697          .LONG   0
                     0004  2698
                     0004  2699 ;
                     0004  2700 ; Define the fields that make up the LOCAL_PORTS class.
                     0004  2701 ;
                     0004  2702
00000080             0004  2703 CLASS           =       CLS$M_LPRT
                     0004  2704
                     0004  2705
                     0004  2706          DEFINE_FIELD    <LOCAL>,<LOCSYSID>,6,<LSYSTEM_ID>,DEC,4,4,,FORMAT_DATA
                     000C  2707          DEFINE_FIELD    <LOCAL>,<NUM_BUF_DSCR>,2,<BUFF_DESCRIPTORS>,-
                     000C  2708                          DEC,2,10,,FORMAT_DATA
                     0014  2709          DEFINE_FIELD    <LOCAL>,<NUM_FREE_BUF_DSCR>,4,-
                     0014  2710                          <FREE_BUFF_DESCRIPTORS>,DEC,2,10,,FORMAT_DATA
                     001C  2711
                     001C  2712
                     001C  2713          DEFINE_FIELD    <LOCAL>,<MB$L_PORT_NAME>,4,<NAME>,ASCII,3,4,1,
                     0024  2714          DEFINE_FIELD    <LOCAL>,<MB$L_STS>,PORT_STATUS_STRNG_LEN,-
                     0024  2715                          <LP_STATUS>,ASCIC,4,6,1,DISPLAY_PORT_STATUS
                     002C  2716          DEFINE_FIELD    <LOCAL>,<MB$L_PORT_NUM>,1,<PORT_NUM>,DEC,1,4,1
                     0034  2717          DEFINE_FIELD    <LOCAL>,<MB$W_DG_FREE>,2,<DGS_FREE>,ASCII,1,3,1
                     003C  2718          DEFINE_FIELD    <LOCAL>,<MB$W_MSG_FREE>,2,<MSGS_FREE>,ASCII,1,3,1
                     0044  2719          DEFINE_FIELD    <LOCAL>,<PDT$T_CNTOWNER>,16,<COUNTER_OWNER>,ASCII,8,16
                     004C  2720          DEFINE_FIELD    <LOCAL>,<MB$L_PL_WAITERS>,4,<POOL_WAITERS>,DEC,2,4
                     0054  2721          DEFINE_FIELD    <LOCAL>,<PDT$L_MSGHDRSZ>,4,<MSG_HDR_SIZE>,DEC,5,6
                     005C  2722          DEFINE_FIELD    <LOCAL>,<PDT$L_DGOVRHD>,4,<DG_OVRHD_SIZE>,DEC,4,6
                     0064  2723          DEFINE_FIELD    <LOCAL>,<MB$L_DEVTYPE>,DEV_TYP_STRNG_LEN,-
                     0064  2724                          <LP_TYPE>,ASCIC,4,5,,DISPLAY_PORT_TYPE
                     006C  2725          DEFINE_FIELD    <LOCAL>,<PDT$W_PBCOUNT>,2,<OPEN_CIRCS>,DEC,3,6,1
                     0074  2726          DEFINE_FIELD    <LOCAL>,<MB$L_FORM_PB>,4,<FORM_CIRCS>,DEC,3,6,1
```

```
                     007C  2727          DEFINE_FIELD    <LOCAL>,<MB$W_CMDS_QUED>,2,<CMDS_QUEUED>,ASCII,2,4
                     0084  2728          DEFINE_FIELD    <LOCAL>,<MB$W_RSPS_QUED>,2,<RSPS_QUEUED>,ASCII,2,4
                     008C  2729          DEFINE_FIELD    <LOCAL>,<PDT$B_MAX_PORT>,1,<MAX_PORT>,DEC,5,8
                     0094  2730          DEFINE_FIELD    <LOCAL>,<PDT$B_PORTMAP>,2,<PRT_MAP>,HEX,5,7
                     009C  2731          DEFINE_FIELD    <LOCAL>,<PDT$B_DGIMAP>,2,<DGI_MAP>,HEX,3,6
                     00A4  2732          DEFINE_FIELD    <LOCAL>,<PDT$B_PLOGMAP>,2,<LOG_MAP>,HEX,5,7
                     00AC  2733          DEFINE_FIELD    <LOCAL>,<PDT$B_PO_LBSTS>,LB_STAT_STRG_LEN,-
                     00AC  2734                          <LB_STATUS>,ASCIC,5,5,,DISPLAY_LOOPBACK_STATUS
                     00B4  2735
                     00B4  2736
                     00B4  2737 ;
                     00B4  2738 ; Define the fields that make up the ERRORS class.
                     00B4  2739 ;
                     00B4  2740
            00000100 00B4  2741 CLASS           =       CLS$M_ERR
                     00B4  2742
                     00B4  2743
                     00B4  2744          DEFINE_FIELD    <LOCAL>,<MB$B_ERTCNT>,1,<ERT_COUNT>,DEC,3,5,1
                     00BC  2745          DEFINE_FIELD    <LOCAL>,<MB$B_ERTMAX>,1,<ERT_MAX>,DEC,5,5,1
                     00C4  2746          DEFINE_FIELD    <LOCAL>,<MB$W_ERRCNT>,1,<NUM_ERRORS>,DEC,6,6,1
                     00CC  2747
                     00CC  2748
                000000CC   2749          .PSECT  LOCAL_KEYTABLE,RD,NOWRT,NOEXE
                     00CC  2750
                     00CC  2751
            00000019 00CC  2752 NUM_LOCAL_DSCRS = KEYWRDVAL             ; Save the number of descriptors in the
                     00CC  2753                                         ;  local report.
                     00CC  2754
            00000000 00CC  2755 . =     LOCAL_KEYTAB                    ; Save the vector count for
            00000032 0000  2756         .LONG   KEYWRDVAL*2             ;  LIB$LOOKUP_KEY.
            0000003A 0004  2757 . =     .+KEYWRDVAL*2
                     003A  2758
                00000000   2759          .PSECT  LOCAL_BITMAP,RD,WRT,NOEXE
                     0000  2760
                     0000  2761 ;
                     0000  2762 ; Save any partial mask in the bitmap and increase the count of the bitmap's
                     0000  2763 ; length.
                     0000  2764 ;
            FFFFFFF8 0000  2765          .IF     NE      COUNT-33
            01C060F8 0000  2766          .LONG   MASK
            00000000 0004  2767         MAP_LENGTH      =       MAP_LENGTH+1
                     0004  2768          .ENDC
                     0004  2769
                     0004  2770
            00000000 0004  2771 LOCAL_MAP_LEN = MAP_LENGTH              ; Save the number of longwords in the
                     0004  2772                                         ;  local bitmap.
                     0004  2773
                     0004  2774 LOC_SAVE_BITMAP:                        ; Reserve enough room for a copy of the
            00000008 0004  2775          .BLKL   MAP_LENGTH+1           ;  local bitmap to be used by the
                     0008  2776                                         ;  INITIALIZE command.
                     0008  2777
                     0008  2778
                000022E4   2779          .RESTORE_PSECT
                     22E4  2780
                     22E4  2781
```

```
                     22E4  2783          .SBTTL  Main program loop.
                 00000000  2784          .PSECT  MAIN,RD,NOWRT,EXE
                     0000  2785 ;++
                     0000  2786 ;
                     0000  2787 ; FUNCTIONAL DESCRIPTION:
                     0000  2788 ;
                     0000  2789 ;       This is the main entry point for show cluster.  The basic program
                     0000  2790 ;       flow is as follows:
                     0000  2791 ;
                     0000  2792 ;       1)      Initialize
                     0000  2793 ;       2)      Capture a snapshot of the SCS database.
                     0000  2794 ;       3)      Display the report.
                     0000  2795 ;       4)      Set a timer and wait for expiration or unsolicited data.
                     0000  2796 ;               (command input or broadcast message received)
                     0000  2797 ;       5)      Process the unsolicited data and/or go to step 2.
                     0000  2798 ;
                     0000  2799 ;--
                     0000  2800
                   0CBC  0000  2801 .ENTRY  START,^M<R2,R3,R4,R5,R7,R10,R11>
                     0002  2802
00000476'EF    09    C8  0002  2803          BISL    #<MF$M_INIT_FLAG!-      ; Set flags to indicate we're performing
                     0009  2804                    MF$M_RIP>,MISC_FLAGS  ;  initialization and that the full
                     0009  2805                                          ;  screen must be drawn.
                0856   30  0009  2806          BSBW    INIT                ; Initialize.
                     000C  2807          JLBC    R0,MAIN_ERR             ; Branch on error.
                     0012  2808
                     0012  2809 MAIN_LOOP:
00002247'EF    03    E0  0012  2810          BBS     #DVT$V_UNKNOWN,OUT_DEV,-; Set up screen buffer on each pass if
                08         0019  2811                  10$                 ;  output device is an unknown terminal
                     001A  2812                                          ;  type.
                00    E1  001A  2813          BBC     #MF$V_INIT_FLAG,-       ; Otherwise set up screen buffer only
      13 00000476'EF         001C  2814                  MISC_FLAGS,20$      ;  during initialization.
         0000055B'EF    7F  0022  2815 10$:     PUSHAQ  SCREEN_DSCR         ; Set up a buffer for the screen.
    00000000'GF    01    FB  0028  2816          CALLS   #1,G^LIB$SET_BUFFER ;
                     002F  2817          JLBC    R0,MAIN_ERR             ; Branch on error.
                     0035  2818
 1F 00002247'EF    01    E1  0035  2819 20$:     BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if output device isn't a
                     003D  2820                                          ;  DECcrt.
          0000224C'EF    DF  003D  2821          PUSHAL  COL_NUM             ; Set line drawing and special graphics
          00002248'EF    DF  0043  2822          PUSHAL  LINE_NUM            ;  character set in G1 area in DECcrt.
          00000168'EF    DF  0049  2823          PUSHAL  GRAPHICS_DSGNTR     ;
    00000000'GF    03    FB  004F  2824          CALLS   #3,G^LIB$PUT_SCREEN ;
                     0056  2825          JLBC    R0,MAIN_ERR             ; Branch on error.
                     005C  2826
                  0F9A   30  005C  2827 30$:     BSBW    DISPLAY             ; Generate the display output.
                     005F  2828          JLBC    R0,MAIN_ERR             ; Branch on error.
    00000000'GF    00    FB  0065  2829          CALLS   #0,G^LIB$PUT_BUFFER ; Output the report to the screen.
    00000476'EF    04    E0  006C  2830          BBS     #MF$V_CONT,MISC_FLAGS,- ; Branch if continuous display.
                22         0073  2831                  40$                 ;
          0000224C'EF    DF  0074  2832          PUSHAL  COL_NUM             ; Otherwise place the cursor at the
          0000225C'EF    DF  007A  2833          PUSHAL  CMD_LINE            ;  bottom of the screen.
    00000000'GF    02    FB  0080  2834          CALLS   #2,G^LIB$SET_CURSOR ;
                     0087  2835          JLBC    R0,MAIN_ERR             ; Branch on error.
                     008D  2836          $EXIT_S                         ; Exit.
                     0096  2837
                     0096  2838 ;
                     0096  2839 ; If input is from a command file, then continute reading commands from the
```

```
                                0096  2840 ; file.  Otherwise, go to sleep until a key in pressed, a broadcast message
                                0096  2841 ; comes in, or the interval timer expires.
                                0096  2842 ;
                                0096  2843
39 00002247'EF   02   E1        0096  2844 40$:    BBC     #DVT$V_HARD,OUT_DEV,50$ ; Branch if not hardcopy terminal.
         0000055B'EF   7F        009E  2845        PUSHAQ  SCREEN_DSCR             ; Set up a buffer for the screen.
   00000000'GF   01   FB        00A4  2846        CALLS   #1,G^LIB$SET_BUFFER     ;
                                00AB  2847        JLBC    R0,MAIN_ERR             ; Branch on error.
1E 00002246'EF   05   E1        00B1  2848        BBC     #DVT$V_DISK,IN_DEV,50$  ; Branch if input isn't from a file.
   00001215'EF   00   FB        00B9  2849        CALLS   #0,EXECUTE              ; Go process command input.
                                00C0  2850        JLBC    R0,MAIN_ERR             ; Branch on error.
                 00   E1        00C6  2851        BBC     #MF$V_INIT_FLAG,-       ; Branch if initialization not in
     09 00000476'EF             00C8  2852                MISC_FLAGS,50$          ;  progress.
                                00CE  2853        $EXIT_S                         ; Otherwise exit.
                                00D7  2854
                                00D7  2855 50$:    $SETIMR_S -                     ; Set the refresh interval timer.
                                00D7  2856                EFN=#EVF$V_TIM_FLAG,-   ;
                                00D7  2857                DAYTIM=DELAY,-          ;
                                00D7  2858                REQIDT=#1               ;
                                00EA  2859        JLBC    R0,MAIN_ERR             ; Branch on error.
                                00F0  2860
                                00F0  2861 ;
                                00F0  2862 ; Wait for expiration of interval timer, or an AST to occur.
                                00F0  2863 ;
                                00F0  2864
                                00F0  2865        $WFLOR_S -
                                00F0  2866                EFN=#0,-
                                00F0  2867                MASK=#EVF$M_TIM_FLAG!EVF$M_AST_FLAG
                                00FB  2868        JLBC    R0,MAIN_ERR             ; Branch on error.
                                0101  2869
                                0101  2870        $CANTIM_S -                     ; Now that we're awake, cancel
                                0101  2871                REQIDT=#1               ;  outstanding refresh timer requests.
                                010C  2872        JLBC    R0,MAIN_ERR             ; Branch on error.
                                0112  2873
                                0112  2874        $READEF_S -                     ; Read flags to determine which
                                0112  2875                EFN=#0,-                ;  condition occured.
                                0112  2876                STATE=FLAG_STATE
                                0121  2877        JLBC    R0,MAIN_ERR             ; Branch on error.
                                0127  2878
                   02   E0      0127  2879        BBS     #EVF$V_AST_FLAG,-       ; Branch if AST flag set.
     21 00000472'EF             0129  2880                FLAG_STATE,70$
00000476'EF   05   E0        012F  2881        BBS     #MF$V_CYCLE,MISC_FLAGS,-; Branch if we're not cycling the
                   03             0136  2882                60$                     ;  display.
                 FED8   31      0137  2883        BRW     MAIN_LOOP               ; Otherwise start over.
          000021AB'EF   96      013A  2884 60$:    INCB    SWITCH                  ; Toggle switch.
                                0140  2885        JLBS    SWITCH,MAIN_LOOP        ; Branch if not time to switch displays.
                 2419   30      014A  2886        BSBW    SWITCH_DISPLAY          ; Otherwise, switch the displays
                 FEC2   31      014D  2887        BRW     MAIN_LOOP               ;  and do a screen update.
                                0150  2888 70$:    $CLREF_S -                      ; Reset AST flag.
                                0150  2889                EFN=#EVF$V_AST_FLAG     ;
                                0159  2890        JLBC    R0,MAIN_ERR             ; Branch on error.
01    00001FB5'EF   B1      015F  2891        CMPW    MBX_BUF,#MSG$_TRMUNSOLIC; Was AST caused by unsolicited data?
                   0E   12      0166  2892        BNEQ    80$                     ; Branch if not.
          00000199'EF   16      0168  2893        JSB     READ_LOOP               ; Otherwise go process the command.
                                016E  2894        JLBC    R0,MAIN_ERR             ; Branch on error.
                   17   11      0174  2895        BRB     MAIN_CONT               ;
0053 8F   00001FB5'EF   B1      0176  2896 80$:    CMPW    MBX_BUF,#MSG$_TRMBRDCST ; Was AST caused by terminal broadcast?
```

```
                     0C   12  017F  2897          BNEQ    MAIN_CONT               ; If not, ignore AST.
0000045B'EF          16   13  0181  2898          JSB     BRDCST                  ; Otherwise, go handle broadcast
                          14  0187  2899                                          ;  message.
                          15  0187  2900          JLBC    R0,MAIN_ERR             ; Branch on error.
                          16  018D  2901
                          17  018D  2902 MAIN_CONT:
                        18A9  30  18D  2903          BSBW    SET_AST                 ; Set up for next AST.
                          19  0190  2904          JLBC    R0,MAIN_ERR             ; Branch on error.
                        FE79  31  0196  2905          BRW     MAIN_LOOP               ; Start again.
                          1A  0199  2906
                          1B  0199  2907          .DSABL  LSB
                          1C  0199  2908
```

```
                     0199  2910 ;++
                     0199  2911 ;
                     0199  2912 ; FUNCTIONAL DESCRIPTION:
                     0199  2913 ;
                     0199  2914 ;       This routine is entered when a unsolicited data AST occurs.  It
                     0199  2915 ;       examines the type-ahead buffer and then issues a read QIO with or
                     0199  2916 ;       without a prompt depending on the character found in the type-ahead
                     0199  2917 ;       buffer.
                     0199  2918 ;
                     0199  2919 ;       Once the read completes control is passed to DCL_PARSE or to
                     0199  2920 ;       the appropriate routine to handle control or escape sequences.
                     0199  2921 ;
                     0199  2922 ; CALLING SEQUENCE:
                     0199  2923 ;
                     0199  2924 ;       JSB     READ_LOOP
                     0199  2925 ;
                     0199  2926 ; CALLED BY:
                     0199  2927 ;
                     0199  2928 ;       MAIN_LOOP
                     0199  2929 ;
                     0199  2930 ;--
                     0199  2931
                     0199  2932         .ENABL  LSB
                     0199  2933
                     0199  2934 READ_LOOP:
                     0199  2935         $QIOW_S -                       ; Get the number of characters in the
                     0199  2936                 CHAN=IN_CHAN,-          ;  type ahead buffer.
                     0199  2937                 FUNC=#IO$_SENSEMODE!IO$M_TYPEAHDCNT,-
                     0199  2938                 P1=TYPEAHD_BUF
                     01BE  2939         JLBC    R0,999$                 ; Branch on error.
          000004BD'EF  B5  01C4  2940         TSTW    TYPEAHD_BUF             ; Are there any characters in the type
                     01CA  2941                                         ;  ahead buffer?
               03    12  01CA  2942         BNEQU   30$                     ; Branch if there are.
             023F    31  01CC  2943         BRW     999$                    ; Return with status.
                     01CF  2944
       1B 000004BF'EF    91  01CF  2945 30$:    CMPB    TYPEAHD_BUF+2,#ESC      ; Is there an escape sequence in the
                     01D6  2946                                         ;  type ahead buffer?
               1F    13  01D6  2947         BEQL    READ_WO_PROMPT          ; Branch if so.
    9B 8F 000004BF'EF    91  01D8  2948         CMPB    TYPEAHD_BUF+2,#CSI      ; Is there an 8 bit escape sequence in
                     01E0  2949                                         ;  the type ahead buffer?
               15    13  01E0  2950         BEQL    READ_WO_PROMPT          ; Branch if so.
       17 000004BF'EF    91  01E2  2951         CMPB    TYPEAHD_BUF+2,#CTRLW    ; Is there a ^W in the type ahead
                     01E9  2952                                         ;  buffer?
               0C    13  01E9  2953         BEQL    READ_WO_PROMPT          ; Branch if so.
       1A 000004BF'EF    91  01EB  2954         CMPB    TYPEAHD_BUF+2,#CTRLZ    ; Is there a ^Z in the type ahead
                     01F2  2955                                         ;  buffer?
                     01F2  2956         JNEQ    READ_W_PROMPT           ; Branch if not.
                     01F7  2957
                     01F7  2958 READ_WO_PROMPT:
                     01F7  2959         $QIOW_S -                       ; Read the command without prompt.
                     01F7  2960                 CHAN=IN_CHAN,-          ;
                     01F7  2961                 FUNC=#IO$M_EXTEND!IO$_READVBLK,-
                     01F7  2962                 IOSB=IOSB,-             ;
                     01F7  2963                 P1=CMD_BUF,-            ;
                     01F7  2964                 P2=#CMD_BUF_LEN,-       ;
                     01F7  2965                 P5=#TERM_LIST,-         ;
                     01F7  2966                 P6=#TERM_LIST_LEN       ;
```

```
                                 022A  2967          JLBC    R0,999$                 ; Branch on error.
                                 0230  2968
            1A  00002242'EF   91 0230  2969          CMPB    IOSB+4,#CTRLZ           ; Did the user type an control-Z?
                          0C  12 0237  2970          BNEQ    40$                     ; Branch if not.
                       50 01  D0 0239  2971          MOVL    #SS$_NORMAL,R0          ; Indicate normal exit.
                              023C  2972          $EXIT_S                         ; Exit
                              0245  2973
          9B 8F   00002242'EF   91 0245  2974 40$:  CMPB    IOSB+4,#CSI             ; Did the user type an eight-bit escape
                                024D  2975                                          ;  sequence?
                          09  13 024D  2976          BEQL    50$                     ; Branch if so.
            1B    00002242'EF   91 024F  2977          CMPB    IOSB+4,#ESC             ; Did the user type an escape sequence?
                          06  12 0256  2978          BNEQ    60$                     ; Branch if not.
                     145E 30 0258  2979 50$:  BSBW    SET_SCROLL              ; Set scroll.
                     FF3B 31 025B  2980          BRW     READ_LOOP               ; Loop till type ahead buffer is empty.
            17    00002242'EF   91 025E  2981 60$:  CMPB    IOSB+4,#CTRLW           ; Did user type control-W?
                                0265  2982          JLBC    R0,999$                 ; Error if not.
                  00000470'EF   94 026B  2983          CLRB    TOP_CMD_LINE            ; Indicate that there is no additional
                                0271  2984                                          ;  information on the top command line.
    00002248'EF   0000225C'EF   D0 0271  2985          MOVL    CMD_LINE,LINE_NUM       ; Get the number of the last line of
                                027C  2986                                          ;  the command display area.
                     217E 30 027C  2987          BSBW    ERASE_TWO_LINES         ; Erase any information in the command
                                027F  2988                                          ;  display area.
                                027F  2989          JLBC    R0,999$                 ; Branch on error.
              00002348'EF   00  FB 0285  2990          CALLS   #0,REFRESH              ; Otherwise, refresh the screen.
                                028C  2991          JLBC    R0,999$                 ; Branch on error.
                     FF04 31 0292  2992          BRW     READ_LOOP               ; Loop till type ahead buffer is empty.
                              0295  2993
                              0295  2994 READ_W_PROMPT:
    00002248'EF   0000225C'EF   D0 0295  2995          MOVL    CMD_LINE,LINE_NUM       ; Set cursor to line 23,
          06 00002246'EF   01  E1 02A0  2996          BBC     #DVT$V_DCRT,IN_DEV,70$  ;  or if terminal is a DECcrt,
                00002248'EF   D6 02A8  2997          INCL    LINE_NUM                ;  on line 24.
                0000224C'EF   3F 02AE  2998 70$:  PUSHAW  COL_NUM                 ;
                00002248'EF   3F 02B4  2999          PUSHAW  LINE_NUM                ;
       00000000'GF    02  FB 02BA  3000          CALLS   #2,G^LIB$SET_CURSOR     ;
                              02C1  3001          JLBC    R0,999$                 ; Branch on error.
       00000458'EF    01  D0 02C7  3002          MOVL    #1,SCROLL_DELAY         ; Set the number of refreshes before
                              02CE  3003                                          ;  the command is scrolled.
          09 000004B9'EF   E8 02CE  3004          BLBS    CMD_STATUS,80$          ; Branch if the last command succeeded.
                     1887 30 02D5  3005          BSBW    SCROLL_TWO              ; Scroll the two command lines.
                              02D8  3006          JLBC    R0,999$                 ; Branch on error.
       00000000'GF    00  FB 02DE  3007 80$:  CALLS   #0,G^LIB$ERASE_LINE     ; Erase the command line.
                              02E5  3008          JLBC    R0,999$                 ; Branch on error.
    0C 00002246'EF    01  E1 02EB  3009          BBC     #DVT$V_DCRT,IN_DEV,90$  ; Skip the following input device isn't
                              02F3  3010                                          ;  a DECcrt.
            000019EA'EF   16 02F3  3011          JSB     SET_REGION_COMMAND      ; Establish a scrolling region.
                              02F9  3012          JLBC    R0,999$                 ; Branch on error.
                              02FF  3013
                              02FF  3014 90$:  $QIOW_S -                       ; Read the command with prompt.
                              02FF  3015                  CHAN=IN_CHAN,-
                              02FF  3016                  FUNC=#IO$_READPROMPT!IO$M_DSABLMBX,-
                              02FF  3017                  IOSB=IOSB,-
                              02FF  3018                  P1=CMD_BUF,-
                              02FF  3019                  P2=#CMD_BUF_LEN,-
                              02FF  3020                  P5=#PROMPT+8,-
                              02FF  3021                  P6=#PROMPT_LEN
                              0332  3022          JLBC    R0,999$                 ; Branch on error.
                              0338  3023
```

```
                1A  00002242'EF   91  0338  3024          CMPB    IOSB+4,#CTRLZ           ; Did user type control-Z?
                          0C      12  033F  3025          BNEQ    100$                    ; Branch if not.
                       50 01      D0  0341  3026          MOVL    #SS$_NORMAL,R0          ; Indicate normal exit.
                                      0344  3027          $EXIT_S                         ; Exit
                                      034D  3028
    000020D3'EF 00002240'EF   3C      034D  3029 100$:    MOVZWL  IOSB+2,CMD_LINE_DSCR    ; Save number of characters in command.
                00020D3'EF    DF      0358  3030          PUSHAL  CMD_LINE_DSCR           ; Convert command line to uppercase.
                00020D3'EF    DF      035E  3031          PUSHAL  CMD_LINE_DSCR           ;
            00000000'GF 02    FB      0364  3032          CALLS   #2,G^STR$UPCASE         ;
                                      036B  3033          JLBC    R0,999$                 ; Branch on error.
                                      0371  3034
          40 8F 000004BF'EF   91      0371  3035          CMPB    TYPEAHD_BUF+2,#^A/@/    ; Is there a "@" in the type ahead
                                      0379  3036                                          ;  buffer?
                          2C  13      0379  3037          BEQL    110$                    ; If so, bypass DCL_PARSE.
                                      037B  3038
                                      037B  3039 ;
                                      037B  3040 ; Use DCL_PARSE to check for obvious syntax errors.
                                      037B  3041 ;
                                      037B  3042
                0000040F'EF   DF      037B  3043          PUSHAL  SECOND_CHANCE
                00000000'EF   DF      0381  3044          PUSHAL  SHOW_CLUSTER$COMMAND_TABLE
                00020D3'EF    DF      0387  3045          PUSHAL  CMD_LINE_DSCR
            00000000'GF 03    FB      038D  3046          CALLS   #3,G^CLI$DCL_PARSE
                       10 50  E8      0394  3047          BLBS    R0,110$                 ; Branch if no error.
            00000470'EF 04    90      0397  3048          MOVB    #CL$M_ERROR,TOP_CMD_LINE; Indicate that there is error
                                      039E  3049                                          ;  information on both command lines.
            00000471'EF 04    90      039E  3050          MOVB    #CL$M_ERROR,BOT_CMD_LINE;
                        59    11      03A5  3051          BRB     130$                    ;
                                      03A7  3052
                                      03A7  3053 ;
                                      03A7  3054 ; Use TPARSE to do the actual parsing and call the action routines.
                                      03A7  3055 ;
                                      03A7  3056
                                      03A7  3057 ;
                                      03A7  3058 ; Form the descriptor for LIB$TPARSE.
                                      03A7  3059 ;
                                      03A7  3060
000004CF'EF 000020D3'EF   B0      03A7  3061 110$:    MOVW    CMD_LINE_DSCR,TPARSE_BLK+TPA$L_STRINGCNT
000004D3'EF 000020DB'EF   DE      03B2  3062          MOVAL   CMD_BUF,TPARSE_BLK+TPA$L_STRINGPTR
            00000000'EF   DF      03BD  3063          PUSHAL  CMD_KEY                 ; Parse the input line.
            00000000'EF   DF      03C3  3064          PUSHAL  CMD_STATE               ;
            000004C7'EF   DF      03C9  3065          PUSHAL  TPARSE_BLK              ;
        00000000'GF 03    FB      03CF  3066          CALLS   #3,G^LIB$TPARSE         ;
        000004B9'EF 50    D0      03D6  3067          MOVL    R0,CMD_STATUS           ; Save the command status for SIGNAL.
                 0F 50    E9      03DD  3068          BLBC    R0,120$                 ; Branch on error.
        00000470'EF 02    90      03E0  3069          MOVB    #CL$M_CMD,TOP_CMD_LINE  ; Indicate that the last command
                                      03E7  3070                                      ;  entered is now scrolled up one.
             00000471'EF  94      03E7  3071          CLRB    BOT_CMD_LINE            ;  and the bottom command line is blank.
                       11 11      03ED  3072          BRB     130$                    ; Continue.
        00000470'EF 02    90      03EF  3073 120$:    MOVB    #CL$M_CMD,TOP_CMD_LINE  ; Indicate that the last command
                                      03F6  3074                                      ;  entered is now scrolled up one.
        00000471'EF 04    90      03F6  3075          MOVB    #CL$M_ERROR,BOT_CMD_LINE; And the bottom command line contains
                                      03FD  3076                                      ;  an error message.
                   115D   30      03FD  3077          BSBW    SIGNAL                  ; Output the error.
                                      0400  3078
                                      0400  3079 ;
                                      0400  3080 ; Before looping, make sure ERR_MSGVEC still has the address of TPARSE_BLK+
```

```
                        0400  3081 ; TPA$L_TOKENCNT.  This may have changed if an ADD or REMOVE class was just
                        0400  3082 ; parsed.
                        0400  3083 ;
                        0400  3084
000004D7'EF   DE        0400  3085 130$:    MOVAL   TPARSE_BLK+TPA$L_TOKENCNT,-
000004A9'EF             0406  3086                  ERR_MSGVEC+12             ;
        FD8B    31      040B  3087          BRW     READ_LOOP                 ; Loop till no more command.
                        040E  3088
                05      040E  3089 999$:    RSB
                        040F  3090
                        040F  3091          .DSABL  LSB
                        040F  3092
```

```
                        040F  3094 ;++
                        040F  3095 ;
                        040F  3096 ; FUNCTIONAL DESCRIPTION:
                        040F  3097 ;
                        040F  3098 ;       This routine is called when there are parameters or qualifier values
                        040F  3099 ;       missing from the input string.  When called, DCL_PARSE has already
                        040F  3100 ;       set up an arguement list to call LIB$GET_INPUT.  The address that
                        040F  3101 ;       DCL has provided for the return string descriptor is saved and the
                        040F  3102 ;       call to LIB$GET_INPUT is made.  When the string is returned it is
                        040F  3103 ;       concatenated with the original command string so that the whole
                        040F  3104 ;       command can be passed to TPARSE.
                        040F  3105 ;
                        040F  3106 ; CALLING SEQUENCE:
                        040F  3107 ;
                        040F  3108 ;       CALLS   SECOND_CHANCE
                        040F  3109 ;
                        040F  3110 ; CALLED BY:
                        040F  3111 ;
                        040F  3112 ;       CLI$DCL_PARSE from READ_W_PROMPT
                        040F  3113 ;
                        040F  3114 ; INPUTS:
                        040F  3115 ;
                        040F  3116 ;       (AP)    Number of arguments.
                        040F  3117 ;       (AP)+4  Address of descriptor for string returned.
                        040F  3118 ;       (AP)+8  Address of prompt descriptor.
                        040F  3119 ;       (AP)+12 Address to store length of string returned.
                        040F  3120 ;
                        040F  3121 ;--
                        040F  3122
                   003C 040F  3123 .ENTRY  SECOND_CHANCE,^M<R2,R3,R4,R5>
           53    04     D0  0411  3124          MOVL    #4,R3                   ; Get address of DCL's param-routine
         52    6C43     DE  0414  3125          MOVAL   (AP)[R3],R2             ; string descriptor.
00000000'GF     6C     FA  0418  3126          CALLG   (AP),G^LIB$GET_INPUT    ; Get missing parameter from user.
            38 50     E9  041F  3127          BLBC    R0,99$                  ; Branch on error.
51    000020D3'EF     3C  0422  3128          MOVZWL  CMD_LINE_DSCR,R1        ; Get number of characters in original
                        0429  3129                                          ;  command line.
53    000020D7'EF     D0  0429  3130          MOVL    CMD_LINE_DSCR+4,R3      ; Get address of command line.
          53    6341     9E  0430  3131          MOVAB   (R3)[R1],R3             ; Get address of last byte in command.
            83    20     90  0434  3132          MOVB    #^A/ /,(R3)+            ; Change <CR> to <SPACE>.
     51    82    01     C1  0437  3133          ADDL3   #1,(R2)+,R1             ; Get the length of the additional
                        043B  3134                                          ;  input string (plus the space).
000020D3'EF     51     A0  043B  3135          ADDW    R1,CMD_LINE_DSCR        ; Save length of combined strings.
 63    00 B2    51     28  0442  3136          MOVC3   R1,@(R2),(R3)           ; Concatenate the two strings.
      000020D3'EF     DF  0447  3137          PUSHAL  CMD_LINE_DSCR           ; Make sure the full line is uppercase.
      000020D3'EF     DF  044D  3138          PUSHAL  CMD_LINE_DSCR           ;
00000000'GF     02     FB  0453  3139          CALLS   #2,G^STR$UPCASE         ;
                        04  045A  3140 99$:    RET                             ; Return with status.
                        045B  3141
                        045B  3142
```

```
                        045B  3144 ;++
                        045B  3145 ;
                        045B  3146 ; FUNCTIONAL DESCRIPTION:
                        045B  3147 ;
                        045B  3148 ;       This routine is entered when a broadcast message AST occurs.  It
                        045B  3149 ;       determines where the end of a line in the broadcast message occurs
                        045B  3150 ;       and then call STORE_MSG to place the broadcast message in a queue
                        045B  3151 ;       to be output to the terminal.  This routine also replaces non-
                        045B  3152 ;       printable characters in the broadcast message with periods (".").
                        045B  3153 ;
                        045B  3154 ; CALLING SEQUENCE:
                        045B  3155 ;
                        045B  3156 ;       JSB     BRDCST
                        045B  3157 ;
                        045B  3158 ; CALLED BY:
                        045B  3159 ;
                        045B  3160 ;       MAIN_LOOP
                        045B  3161 ;
                        045B  3162 ;--
                        045B  3163
          007C 8F   BB  045B  3164 BRDCST: PUSHR   #^M<R2,R3,R4,R5,R6>
    53  00001FCB'EF   DE  045F  3165        MOVAL   MBX_BUF+22,R3           ; Get address of broadcast message.
    52  00001FC9'EF   3C  0466  3166        MOVZWL  MBX_BUF+20,R2           ; Get length of broadcast message.
        54    52      7D  046D  3167 10$:    MOVQ    R2,R4                   ; Save a copy in R4,R5.
                        0470  3168
                        0470  3169 ;
                        0470  3170 ; Scan the message for non-ascii characters.  If a non-ascii character is
                        0470  3171 ; found, substitute a period (".").  Loop until the first carriage return
                        0470  3172 ; following a valid ascii string is found or until the end of the message
                        0470  3173 ; is found.
                        0470  3174 ;
                     CA 0470  3175        BICL    #<MF$M_VALID_MSG!-      ; Clear valid message flag and <CR>
                        0471  3176                MF$M_CR_FOUND>,-        ;  found after valid message flag.
00000476'EF   00000600 8F       0471  3177                MISC_FLAGS              ;
            56    53    D0  047B  3178 20$:    MOVL    R3,R6                   ; Save pointer to broadcast message.
                  2265  30  047E  3179        BSBW    CHECK_ASCII             ; Check for non-ascii characters.
                54 50     E8  0481  3180        BLBS    R0,70$                  ; Branch if entire message contains
                        0484  3181                                            ;  valid ascii characters.
                56    53    D1  0484  3182        CMPL    R3,R6                   ; Were any ascii at all characters
                        0487  3183                                            ;  found?
                   13   13  0487  3184        BEQL    30$                     ; Branch if not.
                   0A   E0  0489  3185        BBS     #MF$V_CR_FOUND,-        ; Branch if the characters found are
              35 00000476'EF       048B  3186                MISC_FLAGS,60$          ;  part of the next line of a multi-
                        0491  3187                                            ;  line message.
                00000400 8F   C8  0491  3188        BISL    #MF$M_VALID_MSG,-       ; Set valid message found flag.
                00000476'EF       0497  3189                MISC_FLAGS              ;
                   52   D7  049C  3190 30$:    DECL    R2                      ; Count down characters remaining in the
                        049E  3191                                            ;  message.
                   38   19  049E  3192        BLSS    70$                     ; Branch if we've reached the end of the
                        04A0  3193                                            ;  message.
                   0D   63  91  04A0  3194        CMPB    (R3),#^X0D              ; Was a carriage return found?
                   13   12  04A3  3195        BNEQ    40$                     ; Branch if not.
                   09   E1  04A5  3196        BBC     #MF$V_VALID_MSG,-       ; Branch if a carriage return has been
              0B 00000476'EF       04A7  3197                MISC_FLAGS,40$          ;  found before any valid ascii
                        04AD  3198                                            ;  characters have been found.
                00000400 8F   C8  04AD  3199        BISL    #MF$M_CR_FOUND,-        ; Indicate a carriage return following
                00000476'EF       04B3  3200                MISC_FLAGS              ;  a valid message has been found.
```

```
                07   63   91   04B8  3201 40$:    CMPB    (R3),#7                 ; Was a BELL found?
                     04   12   04BB  3202         BNEQ    50$                     ; Branch if not.
                     53   D6   04BD  3203         INCL    R3                      ; Leave the BELL in the buffer
                     BA   11   04BF  3204         BRB     20$                     ;  and loop.
                     83   2E   90   04C1  3205 50$:    MOVB    #^A/./,(R3)+            ; Substitute a '.' for the character.
                     B5   11   04C4  3206         BRB     20$                     ; Loop.
                                   04C6  3207
                                   04C6  3208 ;
                                   04C6  3209 ; A carriage return has been found following a string of valid ascii characters
                                   04C6  3210 ; indicating the end of a line in a multi-line message.  Calculate the length
                                   04C6  3211 ; of the line and store it in the message output queue.
                                   04C6  3212 ;
                                   04C6  3213
                     52   54   7D   04C6  3214 60$:    MOVQ    R4,R2                   ; Restore length and address of full
                                   04C9  3215                                     ;  message.
                52   56   53   C3   04C9  3216         SUBL3   R3,R6,R2                ; Get length of this line of the
                                   04CD  3217                                     ;  message.
                     1A   10   04CD  3218         BSB     STORE_MSG               ; Store this line in the message output
                                   04CF  3219                                     ;  queue.
                52   54   52   C3   04CF  3220         SUBL3   R2,R4,R2                ; Get the length of the remainder of
                                   04D3  3221                                     ;  the message in R2.
                     53   56   D0   04D3  3222         MOVL    R6,R3                   ; Get the address of the remainder of
                                   04D6  3223                                     ;  the message in R3.
                     95   11   04D6  3224         BRB     10$                     ; Loop till entire message processed.
                                   04D8  3225
                                   04D8  3226 ;
                                   04D8  3227 ; The end of the message has been found.  Store the last remainder of the
                                   04D8  3228 ; message in the message output queue.
                                   04D8  3229 ;
                                   04D8  3230
                     52   54   7D   04D8  3231 70$:    MOVQ    R4,R2                   ; Restore length and address of message.
                          0C   10   04DB  3232         BSB     STORE_MSG               ; Store the message in the message
                                   04DD  3233                                     ;  output queue.
00000458'EF          01   D0   04DD  3234         MOVL    #1,SCROLL_DELAY         ; Set the delay counter.
                007C 8F   BA   04E4  3235         POPR    #^M<R2,R3,R4,R5,R6>
                          05   04E8  3236         RSB
                                   04E9  3237
                                   04E9  3238
```

```
                        04E9  3240
                        04E9  3241 ;++
                        04E9  3242 ;
                        04E9  3243 ; FUNCTIONAL DESCRIPTION:
                        04E9  3244 ;
                        04E9  3245 ;       Allocate a buffer from the message buffer free queue.  Store a line
                        04E9  3246 ;       from the broadcast message in the buffer and insert the buffer in the
                        04E9  3247 ;       message output queue.
                        04E9  3248 ;
                        04E9  3249 ; CALLING SEQUENCE:
                        04E9  3250 ;
                        04E9  3251 ;       JSB     STORE_MSG
                        04E9  3252 ;
                        04E9  3253 ; CALLED BY:
                        04E9  3254 ;
                        04E9  3255 ;       BRDCST
                        04E9  3256 ;
                        04E9  3257 ; IMPLICIT INPUTS:
                        04E9  3258 ;
                        04E9  3259 ;       MSG_FRE_QUE contains a queue of buffers allocated and available for use.
                        04E9  3260 ;       MSG_OUT_QUE contains a queue of broadcast message lines to be output.
                        04E9  3261 ;
                        04E9  3262 ; INPUTS:
                        04E9  3263 ;
                        04E9  3264 ;       R2 contains the length of the message.
                        04E9  3265 ;       R3 contains the address of the message.
                        04E9  3266 ;
                        04E9  3267 ; IMPLICIT OUTPUTS:
                        04E9  3268 ;
                        04E9  3269 ;       Message line inserted at end of MSG_OUT_QUE
                        04E9  3270 ;
                        04E9  3271 ;--
                        04E9  3272
                        04E9  3273 STORE_MSG:
00000000'EF  00000008'FF  0F  04E9  3274          REMQUE  @MSG_FRE_QUE,BUFADR     ; Get a from the message buffer free
                              04F4  3275                                          ;  queue.
                          16  1C  04F4  3276          BVC     10$                 ; Branch if buffer available.
             00000000'EF      9F  04F6  3277          PUSHAB  BUFADR              ; Allocate a new buffer.
             00000004'EF      9F  04FC  3278          PUSHAB  BUFSIZE             ;
      00000000'GF    02      FB  0502  3279          CALLS   #2,G^LIB$GET_VM     ;
                       21    50  E9  0509  3280          BLBC    R0,99$              ; Branch on error.
                          3C  BB  050C  3281 10$:     PUSHR   #^M<R2,R3,R4,R5>
   54  00000000'EF      08  C1  050E  3282          ADDL3   #8,BUFADR,R4        ; Get the address to store the message.
                    84    52  B0  0516  3283          MOVW    R2,(R4)+            ; Store message length in the buffer.
                64  63    52  28  0519  3284          MOVC3   R2,(R3),(R4)        ; Store the message in the buffer.
                          3C  BA  051D  3285          POPR    #^M<R2,R3,R4,R5>
00000014'FF  00000000'FF  0E  051F  3286          INSQUE  @BUFADR,@MSG_OUT_QUE+4  ; Insert buffer into broadcast message
                              052A  3287                                          ;  output queue.
                    50    01  D0  052A  3288          MOVL    #SS$_NORMAL,R0
                              05  052D  3289 99$:     RSB                         ; Return with status.
                              052E  3290
                              052E  3291
```

```
                     052E  3293
                     052E  3294 ;++
                     052E  3295 ;
                     052E  3296 ; FUNCTIONAL DESCRIPTION:
                     052E  3297 ;
                     052E  3298 ;       This routine performs the allocation and deallocation of variable
                     052E  3299 ;       length buffers used by the SNAPSHOT routine.  No more than two
                     052E  3300 ;       buffers are ever allocated.  Before taking a new snapshot, the
                     052E  3301 ;       buffer used for the last snapshot becomes the "old" buffer.  This
                     052E  3302 ;       is done by simply moving the new buffer descriptor to the old buffer
                     052E  3303 ;       descriptors (see SAVE_OLD routine).  The previous "old" buffer is then
                     052E  3304 ;       deallocated.  What are now the "old" buffer descriptors are used to
                     052E  3305 ;       determine how much space to allocate for the new buffer based on the
                     052E  3306 ;       following:
                     052E  3307 ;
                     052E  3308 ;              OLD                          New
                     052E  3309 ;       ---------------------------  -------------------------------
                     052E  3310 ;          n System Blocks           n+4 System Blocks
                     052E  3311 ;          x Cluster System Blocks   n+4 Cluster System Blocks
                     052E  3312 ;          y Path Blocks             (n+4)*4 Path Blocks
                     052E  3313 ;          z Connection Descriptors  (n+4)*4+z Connection Descriptors
                     052E  3314 ;
                     052E  3315 ;       Where n,x,y, and z are the number of blocks found by the last snapshot
                     052E  3316 ;       routine.  (not the number last allocated)
                     052E  3317 ;
                     052E  3318 ;       Once the size of the buffer is determined, the buffer is allocated and
                     052E  3319 ;       locked in memory along with the fixed length buffers and other data
                     052E  3320 ;       needed by the snapshot routine.  A snapshot is then taken.  After the
                     052E  3321 ;       snapshot, the number of entries of each of the four types shown above
                     052E  3322 ;       is determined and saved for the next buffer allocation.
                     052E  3323 ;
                     052E  3324 ; CALLING SEQUENCE:
                     052E  3325 ;
                     052E  3326 ;       JSB     DO_SNAPSHOT
                     052E  3327 ;
                     052E  3328 ; CALLED BY:
                     052E  3329 ;
                     052E  3330 ;       DISPLAY
                     052E  3331 ;
                     052E  3332 ; OUTPUTS:
                     052E  3333 ;
                     052E  3334 ;--
                     052E  3335
                     052E  3336 DO_SNAPSHOT:
                  3C BB  052E  3337          PUSHR   #^M<R2,R3,R4,R5>
000003E4'EF       02 91  0530  3338 10$:     CMPB    #2,BUF_COUNT            ; Have we allocated two variable length
                     0537  3339                                             ;  buffers?
                  08 13  0537  3340          BEQL    20$                     ; Branch if so.
      000003E4'EF    96  0539  3341          INCB    BUF_COUNT               ; Increase count of buffers allocated.
                  19 11  053F  3342          BRB     30$                     ; Jump to common code.
      00000419'EF    DF  0541  3343 20$:     PUSHAL  OLD_VAR_BUF             ; Free up the old buffer before we
      00000421'EF    DF  0547  3344          PUSHAL  OLD_VAR_BUF_LEN         ;  allocate a new one.
00000000'GF       02 FB  054D  3345          CALLS   #2,G^LIB$FREE_VM        ;
                     0554  3346          JLBC    R0,99$                  ; Branch on error.
                     055A  3347
                02D8 30  055A  3348 30$:     BSBW    SAVE_OLD                ; Replace old descriptors with new
                     055D  3349                                             ;  descriptors.
```

```
                51   000003FB'EF   B0  055D  3350          MOVW    OLD_SB_DSCR+2,R1        ; Get the number of system blocks found
                                        0564  3351                                          ;  in the last snapshot.
                     51      04     A0  0564  3352          ADDW    #4,R1                   ; Increase by four for expansion.
                00000029'EF  51     B0  0567  3353          MOVW    R1,SB_DSCR+2            ; Set number of system blocks that will
                                        056E  3354                                          ;  fit in the buffer.
    00000027'EF    51    0060 8F     A5  056E  3355          MULW3   #SB$C_LENGTH,R1,SB_DSCR ; Set the length of the system block
                                        0578  3356                                          ;  buffer.
                00000031'EF  51     B0  0578  3357          MOVW    R1,CSB_DSCR+2           ; Set the number of cluster system
                                        057F  3358                                          ;  blocks that will fit in the buffer.
    0000002F'EF    51    00AC 8F     A5  057F  3359          MULW3   #CSB$C_LENGTH,R1,-      ; Set the length of the cluster system
                                        0589  3360                  CSB_DSCR                ;  block buffer.
                     51      04     A4  0589  3361          MULW    #4,R1                   ; Get the number of path blocks that
                                        058C  3362                                          ;  will fit in the buffer.
                00000039'EF  51     B0  058C  3363          MOVW    R1,PB_DSCR+2            ; Save in descriptor.
    00000037'EF    51    0054 8F     A5  0593  3364          MULW3   #PB$C_LENGTH,R1,PB_DSCR ; Set the length of the path block
                                        059D  3365                                          ;  buffer.
                51   00000413'EF   A0  059D  3366          ADDW    OLD_CDT_DSCR+2,R1       ; Allow number of connections to expand
                                        05A4  3367                                          ;  by one per path.
                00000041'EF  51     B0  05A4  3368          MOVW    R1,CDT_DSCR+2           ; Save in descriptor.
                     51    008C 8F     A4  05AB  3369          MULW    #SHC$C_CDT_LENGTH,R1    ; Get the length of the connections
                                        05B0  3370                                          ;  buffer.
                0000003F'EF  51     B0  05B0  3371          MOVW    R1,CDT_DSCR             ; Save in descriptor.
     00000419'EF   000003ED'EF      D0  05B7  3372          MOVL    VAR_BUF,OLD_VAR_BUF     ; Save address of previous buffer.
                   000003F5'EF      D0  05C2  3373          MOVL    VAR_BUF_LEN,-           ; Save length of previous buffer.
                   00000421'EF          05C8  3374                  OLD_VAR_BUF_LEN         ;
                                        05CD  3375 ;
                                        05CD  3376 ; Total the number of bytes needed for the variable length buffers.
                                        05CD  3377 ;
                                        05CD  3378
             000003F5'EF    51     3C  05CD  3379          MOVZWL  R1,VAR_BUF_LEN          ; CDT buffer.
    000003F5'EF    00000027'EF      A0  05D4  3380          ADDW    SB_DSCR,VAR_BUF_LEN     ; SB buffer.
                                0B  1E  05DF  3381          BCC     40$
    000003F5'EF    00010000'EF      C0  05E1  3382          ADDL    ^X10000,VAR_BUF_LEN
    000003F5'EF    0000002F'EF      A0  05EC  3383 40$:     ADDW    CSB_DSCR,VAR_BUF_LEN    ; CSB buffer.
                                0B  1E  05F7  3384          BCC     50$
    000003F5'EF    00010000'EF      C0  05F9  3385          ADDL    ^X10000,VAR_BUF_LEN
    000003F5'EF    00000037'EF      A0  0604  3386 50$:     ADDW    PB_DSCR,VAR_BUF_LEN     ; PB buffer.
                                0B  1E  060F  3387          BCC     60$
    000003F5'EF    00010000'EF      C0  0611  3388          ADDL    ^X10000,VAR_BUF_LEN
                                        061C  3389
                                        061C  3390 ;
                                        061C  3391 ; Save the number of entries we can store in each of the buffers.
                                        061C  3392 ; The numbers will be used to determine how many entries were actually
                                        061C  3393 ; stored in each of the buffers.
                                        061C  3394 ;
                                        061C  3395
    000003E5'EF    00000029'EF      B0  061C  3396 60$:     MOVW    SB_DSCR+2,SB_COUNT      ; Number of entrys that can be stored
                                        0627  3397                                          ;  in the system block buffer.
    000003E7'EF    00000031'EF      B0  0627  3398          MOVW    CSB_DSCR+2,CSB_COUNT    ; Number of entrys that can be stored
                                        0632  3399                                          ;  in the cluster system block buffer.
    000003E9'EF    00000039'EF      B0  0632  3400          MOVW    PB_DSCR+2,PB_COUNT      ; Number of entrys that can be stored
                                        063D  3401                                          ;  in the path block buffer.
    000003EB'EF    00000041'EF      B0  063D  3402          MOVW    CDT_DSCR+2,CDT_COUNT    ; Number of entrys that can be stored
                                        0648  3403                                          ;  in the CDT buffer.
                                        0648  3404
                   000003ED'EF      DF  0648  3405          PUSHAL  VAR_BUF                 ; Allocate some buffer space.
                   000003F5'EF      DF  064E  3406          PUSHAL  VAR_BUF_LEN             ;
```

```
                00000000'GF   02    FB  0654  3407          CALLS   #2,G^LIB$GET_VM          ;
                                      065B  3408          JLBC    R0,99$                  ; Branch on error.
     00000047'EF   000003ED'EF    D0  0661  3409          MOVL    VAR_BUF,VAR_BUF_ADRS    ; Save copy of buffer address for
                                      066C  3410                                          ;  snapshot routine.
     000003F5'EF   000003ED'EF    C1  066C  3411          ADDL3   VAR_BUF,VAR_BUF_LEN,-   ; Calculate address of end of buffer.
                   000003F1'EF        0677  3412                  VAR_BUF_END             ;
                                      067C  3413
          00   000003ED'FF   00    2C  067C  3414          MOVC5   #0,@VAR_BUF,#0,-        ; Zero out the variable length snapshot
          000003ED'FF   03F5'8F      0684  3415                  #VAR_BUF_LEN,@VAR_BUF   ;  buffers.
                                      068C  3416
          00   0000004B'EF   00    2C  068C  3417          MOVC5   #0,FIX_BUF,#0,-         ; Zero out the fixed length snapshot
          0000004B'EF   0458'8F      0694  3418                  #FIX_BUF_LEN,FIX_BUF    ;  buffers.
                                      069C  3419
                                      069C  3420 ;
                                      069C  3421 ; Complete the variable buffer descriptors.
                                      069C  3422 ;
                                      069C  3423
     0000002B'EF   000003ED'EF    D0  069C  3424          MOVL    VAR_BUF,SB_DSCR+4       ; Save address of system block buffer.
                51   00000027'EF    3C  06A7  3425          MOVZWL  SB_DSCR,R1              ; Get length of system block buffer.
00000033'EF   0000002B'EF   51    C1  06AE  3426          ADDL3   R1,SB_DSCR+4,CSB_DSCR+4 ; Save address of cluster system block
                                      06BA  3427                                          ;  buffer.
                51   0000002F'EF    3C  06BA  3428          MOVZWL  CSB_DSCR,R1             ; Get length of cluster system block
                                      06C1  3429                                          ;  buffer.
0000003B'EF   00000033'EF   51    C1  06C1  3430          ADDL3   R1,CSB_DSCR+4,PB_DSCR+4 ; Save address of path block buffer.
                51   00000037'EF    3C  06CD  3431          MOVZWL  PB_DSCR,R1              ; Get length of path block buffer.
00000043'EF   0000003B'EF   51    C1  06D4  3432          ADDL3   R1,PB_DSCR+4,CDT_DSCR+4 ; Save address of CDT buffer.
                                      06E0  3433
                                      06E0  3434 ;
                                      06E0  3435 ; Set up buffer pointers.
                                      06E0  3436 ;
     00000017'EF   0000002B'EF    D0  06E0  3437          MOVL    SB_DSCR+4,SB_BUF_PNTR   ; Initialize pointer into system block
                                      06EB  3438                                          ;  buffer.
     0000001B'EF   00000033'EF    D0  06EB  3439          MOVL    CSB_DSCR+4,CSB_BUF_PNTR ; Initialize pointer into cluster
                                      06F6  3440                                          ;  system block buffer.
     0000001F'EF   0000003B'EF    D0  06F6  3441          MOVL    PB_DSCR+4,PB_BUF_PNTR   ; Initialize pointer into path block
                                      0701  3442                                          ;  buffer.
     00000023'EF   00000043'EF    D0  0701  3443          MOVL    CDT_DSCR+4,CDT_BUF_PNTR ; Initialize pointer into connection
                                      070C  3444                                          ;  descriptor buffer.
                                      070C  3445
                                      070C  3446          $LKWSET_S -                     ; Lock variable length snapshot buffers
                                      070C  3447                  INADR=VAR_BUF           ;  in memory.
                                      071D  3448          JLBC    R0,99$                  ; Branch on error.
                                      0723  3449
                                      0723  3450          $LKWSET_S -                     ; Lock fixed snapshot buffers and
                                      0723  3451                  INADR=SNAP_ADRS         ;  snapshot routine in memory.
                                      0734  3452          JLBC    R0,99$                  ; Branch on error.
                                      073A  3453
                                      073A  3454          $CMKRNL_S -                     ; Capture SCS database.
                                      073A  3455                  ROUTIN=SNAPSHOT         ;
          000004A1'EF   50    D0  0749  3456          MOVL    R0,ERR_MSGVEC+4         ; Save snapshot status.
                                      0750  3457
                                      0750  3458          $ULWSET_S -                     ; Unlock snapshot buffers and snapshot
                                      0750  3459                  INADR=SNAP_ADRS         ;  routine pages.
                                      0761  3460          JLBC    R0,99$                  ; Branch on error.
                                      0767  3461
                                      0767  3462          $ULWSET_S -                     ; Unlock variable snapshot buffer pages.
                                      0767  3463                  INADR=VAR_BUF           ;
```

```
                                0778  3464          JLBC    R0,99$                  ; Branch on error.
                                077E  3465
          50   000004A1'EF    D0  077E  3466          MOVL    ERR_MSGVEC+4,R0         ; Restore snapshot status.
                                0785  3467          JLBC    R0,99$                  ; Branch on error.
                                078B  3468 ;
                                078B  3469 ; Calculate the number of entries actually stored in each of the variable
                                078B  3470 ; length buffers by subtacting the number of empty entries remaining from
                                078B  3471 ; the number of entries we started with.
                                078B  3472 ;
                                078B  3473
000003E5'EF   00000029'EF    A3  078B  3474          SUBW3   SB_DSCR+2,SB_COUNT,-    ; Number of system block entries in
                  00000029'EF       0796  3475                  SB_DSCR+2               ;  buffer.
000003E7'EF   00000031'EF    A3  079B  3476          SUBW3   CSB_DSCR+2,CSB_COUNT,-  ; Number of cluster system block
                  00000031'EF       07A6  3477                  CSB_DSCR+2              ;  entries in buffer.
000003E9'EF   00000039'EF    A3  07AB  3478          SUBW3   PB_DSCR+2,PB_COUNT,-    ; Number of path block entries in
                  00000039'EF       07B6  3479                  PB_DSCR+2               ;  buffer.
000003EB'EF   00000041'EF    A3  07BB  3480          SUBW3   CDT_DSCR+2,CDT_COUNT,-  ; Number of CDT entries in buffer.
                  00000041'EF       07C6  3481                  CDT_DSCR+2              ;
                                07CB  3482
                        00    E4  07CB  3483          BBSC    #MF$V_INIT_FLAG,-       ; Old = new if init pass in progress.
              08 00000476'EF       07CD  3484                  MISC_FLAGS,70$          ;
       57 00000476'EF   03    E1  07D3  3485          BBC     #MF$V_RIP,MISC_FLAGS,99$; Or if refresh just performed.
                        58    10  07DB  3486 70$:     BSB     SAVE_OLD                ;
    0000004B'EF   0458'8F    2C  07DD  3487          MOVC5   #FIX_BUF_LEN,FIX_BUF,-  ; Duplicate the fixed length buffers so
                  0458'8F   00      07E6  3488                  #0,#FIX_BUF_LEN,-       ;  we don't display the data in inverse
                   000004A3'EF      07EA  3489                  OLD_FIX_BUF             ;  video.
                                07EF  3490
                                07EF  3491 ;
                                07EF  3492 ; Check if we got the full data base on this snapshot.  If so go display the
                                07EF  3493 ; report.  Otherwise, loop to expand buffers and try again.
                                07EF  3494 ;
                                07EF  3495
000003E5'EF   00000029'EF    B1  07EF  3496          CMPW    SB_DSCR+2,SB_COUNT      ; Did we run out of system block buffer?
                                07FA  3497          JEQLU   10$                     ; Branch if so.
000003E7'EF   00000031'EF    B1  07FF  3498          CMPW    CSB_DSCR+2,CSB_COUNT    ; Did we run out of clustre system
                                080A  3499                                          ;  block buffer?
                                080A  3500          JEQLU   10$                     ; Branch if so.
000003E9'EF   00000039'EF    B1  080F  3501          CMPW    PB_DSCR+2,PB_COUNT      ; Did we run out of path block buffer?
                                081A  3502          JEQLU   10$                     ; Branch if so.
000003EB'EF   00000041'EF    B1  081F  3503          CMPW    CDT_DSCR+2,CDT_COUNT    ; Did we run out of connection
                                082A  3504                                          ;  descriptor buffer?
                                082A  3505          JEQLU   10$                     ; Branch if so.
                                082F  3506
                     50    01   D0  082F  3507          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                                0832  3508
                           5C   BA  0832  3509 99$:     POPR    #^M<R2,R3,R4,R5>
                                05  0834  3510          RSB                             ; Return with status.
                                    0835  3511
                                    0835  3512
```

```
                                0835  3514 ;++
                                0835  3515 ;
                                0835  3516 ; FUNCTIONAL DESCRIPTION:
                                0835  3517 ;
                                0835  3518 ;       This routine performs two functions:
                                0835  3519 ;               1) When called prior to taking a snapshot it is used to move the
                                0835  3520 ;                  newest set of buffer descriptors to the storage area for the
                                0835  3521 ;                  old buffer descriptors before building a new set.
                                0835  3522 ;               2) When called after taking a snapshot it is used to duplicate
                                0835  3523 ;                  the new buffer descriptors so that both sets point to the
                                0835  3524 ;                  same buffers.  This will mean that the CHECK_DIF routine.
                                0835  3525 ;                  will extract the same data when comparing new and old data.
                                0835  3526 ;
                                0835  3527 ; CALLING SEQUENCE:
                                0835  3528 ;
                                0835  3529 ;       JSB     SAVE_OLD
                                0835  3530 ;
                                0835  3531 ; CALLED BY:
                                0835  3532 ;
                                0835  3533 ;       DO_SNAPSHOT
                                0835  3534 ;
                                0835  3535 ;--
                                0835  3536
                                0835  3537 SAVE_OLD:
000003F9'EF    00000027'EF   7D  0835  3538         MOVQ    SB_DSCR,OLD_SB_DSCR     ; Make descriptor for variable length
00000401'EF    0000002F'EF   7D  0840  3539         MOVQ    CSB_DSCR,OLD_CSB_DSCR   ;  buffers used in the last pass point
00000409'EF    00000037'EF   7D  084B  3540         MOVQ    PB_DSCR,OLD_PB_DSCR     ;  to the ones for this pass so we don't
00000411'EF    0000003F'EF   7D  0856  3541         MOVQ    CDT_DSCR,OLD_CDT_DSCR   ;  display the data in inverse video.
                              05  0861  3542         RSB
                                  0862  3543
```

```
                     0862  3545          .SBTTL  Initialization.
                00000862   3546          .PSECT  MAIN,RD,EXE,NOWRT
                     0862  3547 ;++
                     0862  3548 ;
                     0862  3549 ; FUNCTIONAL DESCRIPTION:
                     0862  3550 ;
                     0862  3551 ;       This routine performs the following steps for program initialization.
                     0862  3552 ;
                     0862  3553 ;       - Establishes an exit handler.
                     0862  3554 ;       - Assigns channels to I/O devices.
                     0862  3555 ;       - Determines the device type of the I/O devices and saves a summary of
                     0862  3556 ;         the type.
                     0862  3557 ;       - Translates logical name SYS$NODE.
                     0862  3558 ;       - Saves the system ID of the local system.
                     0862  3559 ;       - Saves a copy of the display bitmap for use by the INITIALIZE command.
                     0862  3560 ;       - Parses the DCL command line for /REPORT=, /CONTINUOUS, /INTERVAL=,
                     0862  3561 ;         /OUTPUT= and sets up accordingly.
                     0862  3562 ;       - Executes the startup initialization file.
                     0862  3563 ;       - Establishes a control-C handler.
                     0862  3564 ;       - Sets the first AST for unsolicited data.
                     0862  3565 ;       - If input is from a terminal:
                     0862  3566 ;               - A mailbox is associated with the terminal.
                     0862  3567 ;               - New terminal characteristics are set up after saving the old
                     0862  3568 ;                 ones for restoration upon exit.
                     0862  3569 ;               - Screen is cleared if terminal is a scope.
                     0862  3570 ;
                     0862  3571 ; CALLING SEQUENCE:
                     0862  3572 ;
                     0862  3573 ;       JSB     INIT
                     0862  3574 ;
                     0862  3575 ; CALLED BY:
                     0862  3576 ;
                     0862  3577 ;       START
                     0862  3578 ;
                     0862  3579 ; OUTPUTS:
                     0862  3580 ;
                     0862  3581 ;       IN_DEV, OUT_DEV Set to reflect the I/O device type.
                     0862  3582 ;       SAVE_CHAR_BUF   Contains the original terminal characteristics.
                     0862  3583 ;       LOCSYSID        Receives the system ID of the local system.
                     0862  3584 ;       CL_SAVE_BITMAP  Receives a copy of the cluster bitmap.
                     0862  3585 ;       LOC_SAVE_BITMAP Reveives a copy of the local bitmap.
                     0862  3586 ;
                     0862  3587 ; RETURN STATUS:
                     0862  3588 ;
                     0862  3589 ;       SS$_NORMAL or failed status returned in R0.
                     0862  3590 ;
                     0862  3591 ;--
                     0862  3592
00FC 8F    BB        0862  3593 INIT:   PUSHR   #^M<R2,R3,R4,R5,R6,R7>
                     0866  3594
                     0866  3595 ;
                     0866  3596 ; Setup an exit handler.
                     0866  3597 ;
                     0866  3598
                     0866  3599          $DCLEXH_S -                     ; Declare exit handler.
                     0866  3600                    DESBLK=EXIT_BLK       ;
                     0873  3601          JLBC    R0,99$                  ; Branch on error.
```

```
                                0879  3602
                                0879  3603          $GETJPI_S -                             ; Get mask of the user's current
                                0879  3604                  ITMLST=JPI_LIST                 ;  privileges.
                                0890  3605          JLBC    R0,99$                          ; Branch on error.
                                0896  3606
00002230'EF    00    E0         0896  3607          BBS     #PRV$V_CMKRNL,CUR_PRIV,-        ; Branch if user has change mode KERNEL
                     0A         089D  3608                  4$                              ;  privilege.
50    00002804 8F    D0         089E  3609          MOVL    #SS$_NOCMKRNL,R0                ; Indicate CMKRNL required.
               066E   31         08A5  3610          BRW     99$                             ; Exit with error status.
                                08A8  3611
                                08A8  3612 4$:      $GETDVI_S -                             ; Get the input device characteristics.
                                08A8  3613                  DEVNAM=IN_DEV_NAME,-            ;
                                08A8  3614                  ITMLST=GET_LIST                 ;
                                08C6  3615          JLBC    R0,99$                          ; Branch on error.
                                08CC  3616
00002208'EF    42 8F    91      08CC  3617          CMPB    #DC$_TERM,DEVICE_CLASS          ; Is input device a terminal?
                     08    13    08D4  3618          BEQL    5$                              ; Branch if so.
7B 00002246'EF    05    E2      08D6  3619          BBSS    #DVT$V_DISK,IN_DEV,30$          ; Input is from a disk file.
                                08DE  3620
        0000223C'EF    DF      08DE  3621 5$:      PUSHAL  MBXCHAN                         ; Assign a channel to the terminal and
        00002234'EF    DF      08E4  3622          PUSHAL  IN_CHAN                         ;  associate a mailbox with it.
        000004B5'EF    DF      08EA  3623          PUSHAL  BUF_QUO                         ;
        000004B1'EF    DF      08F0  3624          PUSHAL  MAX_MSG                         ;
        000000DE'EF    DF      08F6  3625          PUSHAL  IN_DEV_NAME                     ;
00000000'GF    05    FB      08FC  3626          CALLS   #5,G^LIB$ASN_WTH_MBX            ;
                                0903  3627          JLBC    R0,99$                          ; Branch on error.
                                0909  3628
                                0909  3629 ;
                                0909  3630 ; Check the input terminal characteristics and summarize in IN_DEV.
                                0909  3631 ;
0000221C'EF    20000000 8F    D3  0909  3632          BITL    #TT2$M_DECCRT,DEVDEPEND2; Is terminal a DECcrt?
                        09    13  0914  3633          BEQL    7$                              ; Branch if not.
        00002246'EF     02    88  0916  3634          BISB    #DVT$M_DCRT,IN_DEV              ; Set VT100 bit.
                        3A    11  091D  3635          BRB     30$                             ; Jump to common code.
0000220C'EF    00000040 8F    D3  091F  3636 7$:      BITL    #DT$_VT5X,DEVICE_TYPE           ; Is terminal VT52 compatable?
                        09    13  092A  3637          BEQL    10$                             ; Branch if not.
        00002246'EF     01    88  092C  3638          BISB    #DVT$M_VT52,IN_DEV              ; Set VT52 bit.
                        24    11  0933  3639          BRB     30$                             ; Jump to common code.
00002218'EF    00001000 8F    D3  0935  3640 10$:     BITL    #TT$M_SCOPE,DEVDEPEND           ; Is the terminal some type of scope?
                        10    13  0940  3641          BEQL    20$                             ; Branch if not.
        00002246'EF     01    88  0942  3642          BISB    #DVT$M_VT52,IN_DEV              ; Treat terminal as if its a VT52.
        00002246'EF     08    88  0949  3643          BISB    #DVT$M_UNKNOWN,IN_DEV           ; Indicate actual type is unknown.
                        07    11  0950  3644          BRB     30$                             ; Jump to common code.
        00002246'EF     04    88  0952  3645 20$:     BISB    #DVT$M_HARD,IN_DEV              ; Assume terminal is some type of
                                0959  3646                                                  ;  hardcopy device.
                                0959  3647
                                0959  3648 ;
                                0959  3649 ; Get the output terminal characteristics and summarize in OUT_DEV.  Save
                                0959  3650 ; a copy of the characteristics for restoration upon exit.  Set new
                                0959  3651 ; characteristics NOBRDCST and BRDCSTMBX.
                                0959  3652 ;
                                0959  3653
        0000004B'EF     DF      0959  3654 30$:     PUSHAL  OUTPUT_QUAL                     ; Was /OUTPUT specified?
00000000'GF    01    FB      095F  3655          CALLS   #1,G^CLI$PRESENT                ;
                  12 50    E8  0966  3656          BLBS    R0,31$                          ; Branch if so.
                                0969  3657
        00000101'EF     28    0969  3658          MOVC3   NULL_DEV_NAME,-                 ; Otherwise, NL: is the output device.
```

```
                00000109'EF       096F  3659                 NULL_DEV_NAME+8,-       ;
                00000224'EF       0974  3660                 OUT_FILE_SPEC           ;
                          2F  11  0979  3661           BRB    32$                     ; Branch to common code.
                                  097B  3662
                00001DAB'EF   DF  097B  3663 31$:     PUSHAL RET_BUF_DSCR            ; Get the output qualifier.
                00001DA3'EF   DF  0981  3664           PUSHAL GET_VAL_DSCR            ;
                0000004B'EF   DF  0987  3665           PUSHAL OUTPUT_QUAL             ;
            00000000'GF   03  FB  098D  3666           CALLS  #3,G^CLI$GET_VALUE      ;
                                  0994  3667           JLBC   R0,99$                  ; Branch on error.
                                  099A  3668
    00001DB3'EF 00001DAB'EF   28  099A  3669           MOVC3  RET_BUF_DSCR,RET_BUF,-  ; Store the output file string for
                00000224'EF       09A5  3670                 OUT_FILE_SPEC           ;  $PARSE.
                                  09AA  3671
                                  09AA  3672 32$:     $PARSE FAB = OUTFAB            ; Parse the filespec.
                                  09B7  3673           JLBC   R0,99$                  ; Branch on error.
                                  09BD  3674           JBS    #DEV$V_NET,-            ; Branch if network device.
                                  09BD  3675                  OUTFAB+FAB$L_DEV,57$    ;
                                  09CC  3676           JBC    #DEV$V_TRM,-            ; Branch if output device isn't a
                                  09CC  3677                  OUTFAB+FAB$L_DEV,55$    ;  terminal.
                000001D8'EF   9B  09DB  3678           MOVZBW OUTNAM_BLOCK+NAM$T_DVI,-; Build a descriptor for GETDVI.
                00001DAB'EF       09E1  3679                 RET_BUF_DSCR            ;
                00001DAB'EF   28  09E6  3680           MOVC3  RET_BUF_DSCR,-          ;
                000001D9'EF       09EC  3681                 OUTNAM_BLOCK+NAM$T_DVI+1,- ;
                00001DB3'EF       09F1  3682                 RET_BUF                 ;
                                  09F6  3683           $GETDVI_S -                    ; Get the output device characteristics.
                                  09F6  3684                 DEVNAM=RET_BUF_DSCR,-   ;
                                  09F6  3685                 ITMLST=GET_LIST         ;
                      06 50   E8  0A14  3686 33$:     BLBS   R0,34$                  ; Branch if successful.
                   50    01   D0  0A17  3687           MOVL   #SS$_NORMAL,R0          ; Otherwise treat it like a file.
                      0123   31  0A1A  3688           BRW    55$                     ;
                                  0A1D  3689
        00002208'EF   42 8F   91  0A1D  3690 34$:     CMPB   #DC$_TERM,DEVICE_CLASS  ; Is output device a terminal?
                                  0A25  3691           JNEQ   55$                     ; Branch if not.
                                  0A2A  3692
                                  0A2A  3693           $ASSIGN_S -                    ; Assign a channel to the output device.
                                  0A2A  3694                 DEVNAM=OUT_DEV_NAME,-   ;
                                  0A2A  3695                 CHAN=OUT_CHAN           ;
                                  0A3F  3696           JLBC   R0,99$                  ; Branch on error.
                                  0A45  3697
0000221C'EF   20000000 8F   D3  0A45  3698           BITL   #TT2$M_DECCRT,DEVDEPEND2; Is output device a DECcrt?
                          1D  13  0A50  3699           BEQL   35$                     ; Branch if not.
            00002247'EF   02  88  0A52  3700           BISB   #DVT$M_DCRT,OUT_DEV     ; Set VT100 bit.
0000221C'EF   08000000 8F   D3  0A59  3701           BITL   #TT2$M_AVO,DEVDEPEND2   ; Does the DECcrt have the advanced
                                  0A64  3702                                          ;  video option?
                          43  13  0A64  3703           BEQL   50$                     ; Branch if not.
            00002247'EF   10  88  0A66  3704           BISB   #DVT$M_AVO,OUT_DEV      ; Set advanced video bit.
                          3A  11  0A6D  3705           BRB    50$                     ; Jump to common code.
0000220C'EF   00000040 8F   D3  0A6F  3706 35$:     BITL   #DT$_VT5X,DEVICE_TYPE   ; Is output device VT52 compatable?
                          09  13  0A7A  3707           BEQL   40$                     ; Branch if not.
            00002247'EF   01  88  0A7C  3708           BISB   #DVT$M_VT52,OUT_DEV     ; Set VT52 bit.
                          24  11  0A83  3709           BRB    50$                     ; Jump to common code.
00002218'EF   00001000 8F   D3  0A85  3710 40$:     BITL   #TT$M_SCOPE,DEVDEPEND   ; Is the output device some type of
                                  0A90  3711                                          ;  scope?
                          10  13  0A90  3712           BEQL   45$                     ; Branch if not.
            00002247'EF   01  88  0A92  3713           BISB   #DVT$M_VT52,OUT_DEV     ; Treat output device as if its a VT52.
            00002247'EF   08  88  0A99  3714           BISB   #DVT$M_UNKNOWN,OUT_DEV  ; Indicate actual type is unknown.
                          07  11  0AA0  3715           BRB    50$                     ; Jump to common code.
```

```
                00002247'EF   04   88  0AA2  3716 45$:    BISB    #DVT$M_HARD,OUT_DEV      ; Assume output device is some type of
                                  0AA9  3717                                               ;  hardcopy device.
                                  0AA9  3718 ;
                                  0AA9  3719 ; Build the characteristics buffer for setmode so that the terminal
                                  0AA9  3720 ; characteristics can be restored.
                                  0AA9  3721 ;
                                  0AA9  3722
                00002208'EF   90  0AA9  3723 50$:    MOVB    DEVICE_CLASS,-           ; Store the device class.
                00002214'EF       0AAF  3724                  SAVE_CHAR_BUF            ;
                0000220C'EF   90  0AB4  3725          MOVB    DEVICE_TYPE,-            ; Store the device type.
                00002215'EF       0ABA  3726                  SAVE_CHAR_BUF+1          ;
                00002210'EF   B0  0ABF  3727          MOVW    BUFFER_SIZE,-            ; Store the terminal width.
                00002216'EF       0AC5  3728                  SAVE_CHAR_BUF+2          ;
                                  0ACA  3729
    00002260'EF 0000221B'EF   9A  0ACA  3730          MOVZBL  DEVDEPEND+3,LAST_LINE    ; Save the number of the last line
                                  0AD5  3731                                           ;  (bottom command line) on the
                                  0AD5  3732                                           ;  terminal.
                                  0AD5  3733 ;
                                  0AD5  3734 ; Setup new terminal characteristics for output device.
                                  0AD5  3735 ;
                                  0AD5  3736
                00002214'EF   7D  0AD5  3737          MOVQ    SAVE_CHAR_BUF,-          ; Get a copy of the terminal
                00002220'EF       0ADB  3738                  NEW_CHAR_BUF             ;  characteristics for new setup.
                0000221C'EF   D0  0AE0  3739          MOVL    DEVDEPND2,-              ;
                00002228'EF       0AE6  3740                  NEW_CHAR_BUF+8           ;
                                  0AEB  3741
                00020008 8F   C8  0AEB  3742          BISL    #TT$M_NOBRDCST!TT$M_ESCAPE,-
                00002224'EF       0AF1  3743                  NEW_CHAR_BUF+4           ; Set no broadcast bit and escape bit
                                  0AF6  3744                                           ;  in terminal characteristics.
                00000200 8F   CA  0AF6  3745          BICL    #TT$M_WRAP,-             ; Clear wrap bit in terminal
                00002224'EF       0AFC  3746                  NEW_CHAR_BUF+4           ;  characteristics.
                          10  C8  0B01  3747          BISL    #TT2$M_BRDCSTMBX,-       ; Set broadcast mailbox bit in extended
                00002228'EF       0B03  3748                  NEW_CHAR_BUF+8           ;  terminal characteristics.
                                  0B08  3749
                                  0B08  3750          $QIOW_S -                        ; Set new characteristics.
                                  0B08  3751                  CHAN=OUT_CHAN,-          ;
                                  0B08  3752                  FUNC=#IO$_SETMODE,-      ;
                                  0B08  3753                  P1=NEW_CHAR_BUF,-        ;
                                  0B08  3754                  P2=#CHAR_BUF_LEN         ;
                                  0B2D  3755          JLBC    R0,99$                   ; Branch on error.
                00000080 8F   C8  0B33  3756          BISL    #MF$M_NEW_CHAR,-         ; Indicate that the terminal
                00000476'EF       0B39  3757                  MISC_FLAGS               ;  characteristics have been changed.
                                  0B3E  3758
                          34  11  0B3E  3759          BRB     60$                      ; Jump to common code.
                                  0B40  3760
                00000274'EF   DF  0B40  3761 55$:    PUSHAL  OUT_EXP_DSCR             ; Redirect output to file.
                00002238'EF   DF  0B46  3762          PUSHAL  SET_CHAN                 ;
            00000000'GF   02  FB  0B4C  3763          CALLS   #2,G^LIB$SET_OUTPUT      ;
                                  0B53  3764          JLBC    R0,99$                   ; Branch on error.
                                  0B59  3765
                          24  88  0B59  3766 57$:    BISB    #DVT$M_DISK!DVT$M_HARD,-; Output is to a file.
                00002247'EF       0B5B  3767                  OUT_DEV                  ;
      00002222'EF   0084 8F   B0  0B60  3768          MOVW    #132,SCREEN_WIDTH        ; Assume 132 columns.
    00002260'EF   00000044 8F   D0  0B69  3769          MOVL    #68,LAST_LINE            ; Assume 68 lines.
                                  0B74  3770
0000225C'EF 00002260'EF   01  C3  0B74  3771 60$:    SUBL3   #1,LAST_LINE,CMD_LINE    ; Save the number of the next-to-last
                                  0B80  3772                                           ;  line (top command line) on the
```

```
                                0B80  3773                                    ;  terminal.
                00002260'EF   02  C3  0B80  3774          SUBL3   #2,LAST_LINE,-          ; Save the number of the last line on
                     00002258'EF      0B87  3775                  LAST_DATA_LINE        ;  the terminal to be used to display
                                0B8C  3776                                    ;  data.
                                0B8C  3777
                                0B8C  3778 ;
                                0B8C  3779 ; Save local system ID in LOCSYSID.
                                0B8C  3780 ;
                                0B8C  3781
    00000005'EF   00000000'GF   D0  0B8C  3782 63$:    MOVL    G^SCS$GB_SYSTEMID,LOCSYSID
    00000009'EF   00000004'GF   B0  0B97  3783         MOVW    G^SCS$GB_SYSTEMID+4,LOCSYSID+4
                                0BA2  3784
                                0BA2  3785 ;
                                0BA2  3786 ; Get the local node name.
                                0BA2  3787 ;
                                0BA2  3788
                                0BA2  3789         $CMKRNL_S -
                                0BA2  3790                 ROUTIN=GET_NODE_NAME
                                0BB1  3791
                                0BB1  3792 ;
                                0BB1  3793 ; Process command line qualifier /REPORT=.
                                0BB1  3794 ;
                     0000001B'EF   DF  0BB1  3795         PUSHAL  REPORT_QUAL           ; Was the report name specified?
                00000000'GF   01   FB  0BB7  3796         CALLS   #1,G^C[I$PRESENT      ;
                                0BBE  3797         JLBC    R0,80$                ; If not default to cluster report.
                     00001DA3'EF   DF  0BC4  3798         PUSHAL  GET_VAL_DSCR          ; Get the report name.
                     0000001B'EF   DF  0BCA  3799         PUSHAL  REPORT_QUAL           ;
                00000000'GF   02   FB  0BD0  3800         CALLS   #2,G^C[I$GET_VALUE    ;
                                0BD7  3801         JLBC    R0,80$                ; Default to cluster report on error.
          4C 8F   00001DB3'EF   91  0BDD  3802         CMPB    RET_BUF,#^A/L/        ; Local report requested?
                              25   13  0BE5  3803         BEQL    75$                   ; Yes go setup for the local report.
          4C43 8F   00001DB3'EF   B1  0BE7  3804         CMPW    RET_BUF,#^A/CL/       ; Cluster report requested?
                              61   13  0BF0  3805         BEQL    80$                   ; Yes, go setup for the cluster report.
          5943 8F   00001DB3'EF   B1  0BF2  3806         CMPW    RET_BUF,#^A/CY/       ; /REPORT=CYCLE specified?
                              09   12  0BFB  3807         BNEQ    72$                   ; Branch if not.
                              30   C8  0BFD  3808         BISL    #MF$M_CONT!MF$M_CYCLE,- ; Set cycle and continuous display
                     00000476'EF      0BFF  3809                 MISC_FLAGS            ;  flags.
                              4D   11  0C04  3810         BRB     80$                   ; Set up to start cycling with the
                                0C06  3811                                    ;  cluster report.
                           50   14   D0  0C06  3812 72$:    MOVL    #SS$_BADPARAM,R0      ; Exit status is bad parameter value.
                              0336   31  0C09  3813         BRW     MAIN_ERR              ; Exit.
                                0C0C  3814
                                0C0C  3815
                                0C0C  3816 ;
                                0C0C  3817 ; Setup for the local report.
                                0C0C  3818 ;
                                0C0C  3819
                     00370006 8F   D0  0C0C  3820 75$:    MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
                     00002204'EF      0C12  3821                 CLUST_SAVE
    00000001'EF   00000180 8F   D0  0C17  3822         MOVL    #CLS$M_LPRT!CLS$M_ERR,CLASS_SELECT
                        01   8A  0C22  3823         BICB    #RPT$M_CLSTR,-        ; Set current report to local.
                00000000'EF      0C24  3824                 CURRENT_REPORT
    00002274'EF   00000000'EF   DE  0C29  3825         MOVAL   LOCAL_KEYTAB,KEYTAB_ADRS; Store address of local status keyword
                                0C34  3826                                    ;  table.
    00002264'EF   00000000'EF   DE  0C34  3827         MOVAL   LOCAL_TABLE,ADRS_TABLE  ; Store address of local address table.
    00002268'EF   00000000'EF   DE  0C3F  3828         MOVAL   LOCAL_BITMAP,BITMAP_ADRS; Store address of local bitmap.
             0000226C'EF   00   D0  0C4A  3829         MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Store length of local bitmap.
```

```
                     45   11  0C51  3830          BRB     85$                     ; Branch to common code.
                                0C53  3831
                                0C53  3832 ;
                                0C53  3833 ; Setup for the cluster report.
                                0C53  3834 ;
                                0C53  3835
                 00370006 8F    D0  0C53  3836 80$:    MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
                 00000001'EF        0C59  3837          CLASS_SELECT
                 00000180 8F    D0  0C5E  3838          MOVL    #CLS$M_LPRT!CLS$M_ERR,-
                 00002200'EF        0C64  3839          LOCAL_SAVE
                       01    88  0C69  3840          BISB    #RPT$M_CLSTR,-          ; Set current report to clusters.
                 00000000'EF        0C6B  3841          CURRENT_REPORT          ;
                 00000000'EF    DE  0C70  3842          MOVAL   CLUSTER_KEYTAB,-        ; Store address of cluster keyword
                 00002274'EF        0C76  3843          KEYTAB_ADRS             ;  table.
     00002264'EF 00000000'EF    DE  0C7B  3844          MOVAL   CLUSTER_TABLE,ADRS_TABLE; Store address of cluster table.
                 00000000'EF    DE  0C86  3845          MOVAL   CLUSTER_BITMAP,-        ; Store address of cluster bitmap.
                 00002268'EF        0C8C  3846          BITMAP_ADRS             ;
               0000226C'EF      02  D0  0C91  3847          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Store length of cluster bitmap.
                                0C98  3848
                                0C98  3849 ;
                                0C98  3850 ; Check for /CONTINUOUS.
                                0C98  3851 ;
                                0C98  3852
                     00000029'EF    DF  0C98  3853 85$:    PUSHAL  CONTINUOUS_QUAL         ; Was /CONTINUOUS specified?
                00000000'GF   01    FB  0C9E  3854          CALLS   #1,G^CLI$PRESENT        ;
                        07 50       E9  0CA5  3855          BLBC    R0,87$                  ; Branch if not.
                00000476'EF   10    C8  0CA8  3856          BISL    #MF$M_CONT,MISC_FLAGS   ; Set continuous display flag.
                                0CAF  3857
                                0CAF  3858 ;
                                0CAF  3859 ; Check for /INTERVAL.
                                0CAF  3860 ;
                                0CAF  3861
                     0000003B'EF    DF  0CAF  3862 87$:    PUSHAL  INTERVAL_QUAL           ; Was /INTERVAL specified?
                00000000'GF   01    FB  0CB5  3863          CALLS   #1,G^CLI$PRESENT        ;
                        56 50       E9  0CBC  3864          BLBC    R0,88$                  ; Branch if not.
                00000476'EF   10    C8  0CBF  3865          BISL    #MF$M_CONT,MISC_FLAGS   ; Set continuous display flag.
                     00001DAB'EF    DF  0CC6  3866          PUSHAL  RET_BUF_DSCR            ; Get the time string.
                     00001DA3'EF    DF  0CCC  3867          PUSHAL  GET_VAL_DSCR            ;
                     0000003B'EF    DF  0CD2  3868          PUSHAL  INTERVAL_QUAL           ;
                00000000'GF   03    FB  0CD8  3869          CALLS   #3,G^CLI$GET_VALUE      ;
                                0CDF  3870          JLBC    R0,99$                  ; Branch on error.
                     000021AC'EF    DF  0CE5  3871          PUSHAL  DELAY                   ; Convert the desired number of seconds
                     00001DB3'EF    DF  0CEB  3872          PUSHAL  RET_BUF                 ;  to a longword.
                     00001DAB'EF    DD  0CF1  3873          PUSHL   RET_BUF_DSCR            ;
                00000000'GF   03    FB  0CF7  3874          CALLS   #3,G^LIB$CVT_DTB        ;
                                0CFE  3875          JLBC    R0,99$                  ; Branch on error.
00   000021AC'EF   FF676980 8F    7A  0D04  3876          EMUL    #-10000000,DELAY,#0,-   ; Convert the number of seconds to
                 000021AC'EF        0D10  3877          DELAY                   ;  binary time.
                                0D15  3878
                     000019DC'EF    16  0D15  3879 88$:    JSB     SET_REGION_DATA         ; Set up scrolling region.
                                0D1B  3880
                                0D1B  3881 ;
                                0D1B  3882 ; Save a copy of the cluster bitmap for INITIALIZE command.
                                0D1B  3883 ;
                                0D1B  3884
                          55    02  DU  0D1B  3885 90$:    MOVL    #CLUSTER_MAP_LEN,R5
                57   00000000'EF    DE  0D1E  3886          MOVAL   CLUSTER_BITMAP,R7
```

```
                56   0000000C'EF    DE  0D25  3887          MOVAL   CL_SAVE_BITMAP,R6
                    86    87        D0  0D2C  3888 91$:      MOVL    (R7)+,(R6)+
                      FA 55         F4  0D2F  3889          SOBGEQ  R5,91$
                                        0D32  3890
                                        0D32  3891 ;
                                        0D32  3892 ; Save a copy of the local ports bitmap for INITIALIZE command.
                                        0D32  3893 ;
                        55    00      D0  0D32  3894          MOVL    #LOCAL_MAP_LEN,R5
                57   00000000'EF      DE  0D35  3895          MOVAL   LOCAL_BITMAP,R7
                56   00000004'EF      DE  0D3C  3896          MOVAL   LOC_SAVE_BITMAP,R6
                    86    87        D0  0D43  3897 92$:      MOVL    (R7)+,(R6)+
                      FA 55         F4  0D46  3898          SOBGEQ  R5,92$
                                        0D49  3899
                                        0D49  3900
                                        0D49  3901 ;
                                        0D49  3902 ; Set the initial number of system blocks and connections that can be handled
                                        0D49  3903 ; by the snapshot routine.  These numbers will be used in the MAIN_LOOP routine
                                        0D49  3904 ; to determine the size of buffers to allocate for system blocks and
                                        0D49  3905 ; connections.  The size of buffers for circuits and cluster system blocks will
                                        0D49  3906 ; be derived from the number of system blocks.
                                        0D49  3907 ;
                                        0D49  3908 ; The initial number of connections is set to the number of connections allowed
                                        0D49  3909 ; by the SYSGEN SCSCONNCNT parameter but not to exceed the parameter default
                                        0D49  3910 ; (DEF_CONNCNT).  This is to protect from the possibility of the parameter
                                        0D49  3911 ; being set much higher than necessary. (maximum is 32k)
                                        0D49  3912 ;
                                        0D49  3913 ; The initial number of system blocks is set to 12 which is in affect
                                        0D49  3914 ; the same as setting it to 16 because it will be increased by 4 on the first
                                        0D49  3915 ; pass to allow for expansion.
                                        0D49  3916 ;
          00000000'GF    B1            0D49  3917          CMPW    G^SCS$GW_CDTCNT,-       ; Are there less connections allowed
                    12                  0D4F  3918                  #DEF_CONNCNT            ;  on this system then there normally
                                        0D50  3919                                          ;  are by default?
                    09    1F            0D50  3920          BLSSU   93$                     ; Branch if so.
00000041'EF         12    80            0D52  3921          MOVW    #DEF_CONNCNT,CDT_DSCR+2 ; Otherwise take the default.
                    0B    11            0D59  3922          BRB     94$                     ; Jump to common code.
          00000000'GF    B0            0D5B  3923 93$:      MOVW    G^SCS$GW_CDTCNT,-       ; Store the actual number allowed.
          00000041'EF                  0D61  3924                  CDT_DSCR+2              ;
00000029'EF         0C    B0            0D66  3925 94$:      MOVW    #12,SB_DSCR+2           ; Fake the number of known system
                                        0D6D  3926                                          ;  blocks.  This will be expanded to
                                        0D6D  3927                                          ;  16 on the first pass.
                                        0D6D  3928
                                        0D6D  3929 ;
                                        0D6D  3930 ; Clear the terminals screen.
                                        0D6D  3931 ;
                      1660    30        0D6D  3932          BSBW    CLEAR_SCREEN            ; Clear the screen.
                                        0D70  3933          JLBC    R0,99$                  ; Branch on error.
                                        0D76  3934
                                        0D76  3935 ;
                                        0D76  3936 ; Translate SHOW_CLUSTER$INIT.  If the logical name is defined then go execute
                                        0D76  3937 ; the initialization file.
                                        0D76  3938 ;
                                        0D76  3939
                                        0D76  3940          $TRNLOG_S -                     ; Translate the logical name
                                        0D76  3941                  LOGNAM=INIT_NAME,-      ;  SHOW_CLUSTER$INIT.
                                        0D76  3942                  RSLLEN=TRAN_DSCR,-      ;
                                        0D76  3943                  RSLBUF=TRAN_DSCR        ;
```

```
                50   00000629 8F    D1  0D95  3944          CMPL    #SS$_NOTRAN,R0          ; Was SHOW_CLUSTER$INIT defined?
                                31    13  0D9C  3945          BEQL    95$                     ; Branch if not.
                     00000001'EF    DE  0D9E  3946          MOVAL   DEF_EXT,-               ; Store default file extention in FAB.
                     0000004C'EF        0DA4  3947                  INFAB+FAB$L_DNA         ;
                                04    90  0DA9  3948          MOVB    #DEF_EXT_LEN,-          ; Store length of default extention in
                     00000051'EF        0DAB  3949                  INFAB+FAB$B_DNS         ;  FAB.
            00001215'EF         00    FB  0DB0  3950          CALLS   #0,EXECUTE              ; Execute initialization file.
                                      0DB7  3951          JLBC    R0,99$                  ; Branch on error.
                     00000005'EF    DE  0DBD  3952          MOVAL   DEF_NAME2,-             ; Store default filename in FAB.
                     0000004C'EF        0DC3  3953                  INFAB+FAB$L_DNA         ;
                                16    90  0DC8  3954          MOVB    #DEF_NAME2_LEN,-        ; Store length of default filename in
                     00000051'EF        0DCA  3955                  INFAB+FAB$B_DNS         ;  FAB.
                                      0DCF  3956
                     00000059'EF    DF  0DCF  3957 95$:     PUSHAL  ENDING_QUAL             ; Was /ENDING specified?
            00000000'GF    01    FB  0DD5  3958          CALLS   #1,G^CLI$PRESENT        ;
                       55   50    E9  0DDC  3959          BLBC    R0,96$                  ; Branch if not.
                     00001DAB'EF    DF  0DDF  3960          PUSHAL  RET_BUF_DSCR            ; Get the time string.
                     00001DA3'EF    DF  0DE5  3961          PUSHAL  GET_VAL_DSCR            ;
                     00000059'EF    DF  0DEB  3962          PUSHAL  ENDING_QUAL             ;
            00000000'GF    03    FB  0DF1  3963          CALLS   #3,G^CLI$GET_VALUE      ;
                                      0DF8  3964          JLBC    R0,99$                  ; Branch on error.
                                      0DFE  3965
                     000021B4'EF    7F  0DFE  3966          PUSHAQ  A_TIME                  ; Convert the time string to binary.
                     00001DAB'EF    DF  0E04  3967          PUSHAL  RET_BUF_DSCR            ;
            00000000'GF    02    FB  0E0A  3968          CALLS   #2,G^LIB$CVT_TIME       ;
                                      0E11  3969          JLBC    R0,99$                  ; Branch on error.
                                      0E17  3970
                                      0E17  3971          $SETIMR_S -                     ; Set the timer to force a control-C
                                      0E17  3972                  DAYTIM = A_TIME,-       ;  when it expires.
                                      0E17  3973                  ASTADR = CNTRLC_AST_DEL ;
                                      0E2E  3974          JLBC    R0,99$                  ; Branch on error.
                                      0E34  3975
                     00000067'EF    DF  0E34  3976 96$:     PUSHAL  BEGIN_QUAL              ; Was /BEGINNING specified?
            00000000'GF    01    FB  0E3A  3977          CALLS   #1,G^CLI$PRESENT        ;
                       05   50    E8  0E41  3978          BLBS    R0,97$                  ; Branch if so.
                     50    01    D0  0E44  3979          MOVL    #SS$_NORMAL,R0          ; Otherwise, make sure we don't
                                      0E47  3980                                          ;  consider the error fatal.
                                79    11  0E47  3981          BRB     975$                    ; Branch to common.
                     00001DAB'EF    DF  0E49  3982 97$:     PUSHAL  RET_BUF_DSCR            ; Get the time string.
                     00001DA3'EF    DF  0E4F  3983          PUSHAL  GET_VAL_DSCR            ;
                     00000067'EF    DF  0E55  3984          PUSHAL  BEGIN_QUAL              ;
            00000000'GF    03    FB  0E5B  3985          CALLS   #3,G^CLI$GET_VALUE      ;
                                      0E62  3986          JLBC    R0,99$                  ; Branch on error.
                                      0E68  3987
                     000021B4'EF    7F  0E68  3988          PUSHAQ  A_TIME                  ; Convert the time string to binary.
                     00001DAB'EF    DF  0E6E  3989          PUSHAL  RET_BUF_DSCR            ;
            00000000'GF    02    FB  0E74  3990          CALLS   #2,G^LIB$CVT_TIME       ;
                                      0E7B  3991          JLBC    R0,99$                  ; Branch on error.
                                      0E81  3992
                                      0E81  3993          $SCHDWK_S -                     ; Leave a wakeup call.
                                      0E81  3994                  DAYTIM = A_TIME         ;
                                7F   50    E9  0E94  3995          BLBC    R0,99$                  ; Branch on error.
                                      0E97  3996
                     00001DAB'EF    DE  0E97  3997          MOVAL   RET_BUF_DSCR,-          ; Save address of time string in error
                     000004A9'EF        0E9D  3998                  ERR_MSGVEC+12           ;  message vector.
00000489'EF  00000000'8F    D0  0EA2  3999          MOVL    #SHOW$_HIBER,CMD_STATUS ; Tell the user we're going into
                              06AD    30  0EAD  4000          BSBW    SIGNAL                  ;  hibernation.
```

```
                     DE  0EB0  4001          MOVAL   <TPARSE_BLK+-                ; Restore error message vector to
                         0EB1  4002                  TPA$L_STRINGCNT>,-          ;  normal.
000004A9'EF  000004CF'EF 0EB1  4003                  ERR_MSGVEC+12               ;
                         0EBB  4004
                         0EBB  4005          $HIBER_S                            ; Off to sleep...
                         0EC2  4006
      03 00002246'EF  05 E1  0EC2  4007 975$:   BBC     #DVT$V_DISK,IN_DEV,98$  ; Branch if input isn't from a file.
                   0049  31  0ECA  4008         BRW     99$                     ; And finish up.
                         0ECD  4009
                         0ECD  4010 ;
                         0ECD  4011 ; Set up control-c handler.
                         0ECD  4012 ;
                         0ECD  4013
                         0ECD  4014 98$:   $QIOW_S -                           ; Setup a control-C ast handler.
                         0ECD  4015                  CHAN=IN_CHAN,-              ;
                         0ECD  4016                  FUNC=#IO$_SETMODE!IO$M_CTRLCAST,-
                         0ECD  4017                  P1=CNTRLC_AST_DEL           ;
                         0EF2  4018
0000054F'EF  00002222'EF 3C  0EF2  4019          MOVZWL  SCREEN_WIDTH,VALUE      ; Save number of columns for set screen
      00000FBA'EF  00 FB  0EFD  4020          CALLS   #0,SET_SCREEN           ; Set the screen width to 80 or 132
                         0F04  4021                                          ;  column mode.
                   0F 50 E9  0F04  4022          BLBC    R0,99$                  ; Branch on error.
                         0F07  4023
                         0F07  4024 ;
                         0F07  4025 ; Set up for first AST.
                         0F07  4026 ;
                         0F07  4027
                         0F07  4028          $CLREF_S -                          ; Clear ast event flag.
                         0F07  4029                  EFN=#EVF$V_AST_FLAG
                   03 50 E9  0F10  4030          BLBC    R0,99$                  ; Branch on error.
                    0B23 30  0F13  4031          BSBW    SET_AST                 ; Set first AST for unsolicited data.
                         0F16  4032
                 C0FC 8F BA  0F16  4033 99$:   POPR    #^M<R2,R3,R4,R5,R6,R7>
                         05  0F1A  4034         RSB                             ; Return with status.
                         0F1B  4035
                         0F1B  4036
```

```
                     0F1B  4038 ;++
                     0F1B  4039 ;
                     0F1B  4040 ; FUNCTIONAL DESCRIPTION:
                     0F1B  4041 ;
                     0F1B  4042 ; This routine simply gets the node name of the local system from the local
                     0F1B  4043 ; system block and stores it in the NODE_NAME_DSCR.  The nodename is then used
                     0F1B  4044 ; by the GET_HEADER routine to display the nodename in the header.
                     0F1B  4045 ;
                     0F1B  4046 ; CALLING SEQUENCE:
                     0F1B  4047 ;
                     0F1B  4048 ;       CALLS   #0,GET_NODE_NAME
                     0F1B  4049 ;
                     0F1B  4050 ; CALLED BY:
                     0F1B  4051 ;
                     0F1B  4052 ;       INIT
                     0F1B  4053 ;
                     0F1B  4054 ; INPUTS:
                     0F1B  4055 ;
                     0F1B  4056 ;       Nodename stored in the local system block.
                     0F1B  4057 ;
                     0F1B  4058 ; OUTPUTS:
                     0F1B  4059 ;
                     0F1B  4060 ;       Nodename stored in NODE_NAME_DSCR.
                     0F1B  4061 ;
                     0F1B  4062 ; RETURN STATUS:
                     0F1B  4063 ;
                     0F1B  4064 ;       None.
                     0F1B  4065 ;
                     0F1B  4066 ;--
                     0F1B  4067
                003C 0F1B  4068 .ENTRY  GET_NODE_NAME,^M<R2,R3,R4,R5>
                     0F1D  4069
                     0F1D  4070 ;
                     0F1D  4071 ; Get the number of bytes in the nodename.
                     0F1D  4072 ;
00000044'GF     90   0F1D  4073         MOVB    G^SCS$GA_LOCALSB+SB$T_NODENAME,-
00000448'EF          0F23  4074                 NODE_NAME_DSCR
                     0F28  4075 ;
                     0F28  4076 ; Store the node name in the node name descriptor.
                     0F28  4077 ;
00000448'EF     28   0F28  4078         MOVC3   NODE_NAME_DSCR,-
00000045'GF          0F2E  4079                 G^SCS$GA_LOCALSB+SB$T_NODENAME+1,-
00000450'EF          0F33  4080                 NODE_NAME
                     0F38  4081
                04   0F38  4082         RET
                     0F39  4083
                     0F39  4084
```

```
                                0F39  4086          .SBTTL  Exit handler.
                                0F39  4087 ;
                                0F39  4088 ;       This is the entry point for the exit routine established during
                                0F39  4089 ;       initialization.  It simply moves the status stored in STATUS by
                                0F39  4090 ;       the $EXIT_S to R0 and falls through to the MAIN_ERR routine.
                                0F39  4091 ;
                                0F39  4092
                         0000   0F39  4093 .ENTRY  EXIT_RTN,^M<>
                                0F3B  4094
      50   00000491'EF    D0   0F3B  4095          MOVL    STATUS,R0               ; Get the exit status.
                                0F42  4096
                                0F42  4097 ;++
                                0F42  4098 ;
                                0F42  4099 ; FUNCTIONAL DESCRIPTION:
                                0F42  4100 ;
                                0F42  4101 ;       This routine handles all cleanup before the program exits.  It restores
                                0F42  4102 ;       the original terminal characteristics, restores the original screen
                                0F42  4103 ;       width, and deassigns the channels assigned to the terminal and mailbox.
                                0F42  4104 ;       An error encountered while trying to do the cleanup is considered fatal
                                0F42  4105 ;       and the image immediately exits.
                                0F42  4106 ;
                                0F42  4107 ; INPUTS:
                                0F42  4108 ;
                                0F42  4109 ;       Error status in R0.
                                0F42  4110 ;
                                0F42  4111 ; IMPLICIT OUTPUTS:
                                0F42  4112 ;
                                0F42  4113 ;       Error message output to terminal if status in R0 is not success.
                                0F42  4114 ;
                                0F42  4115 ; RETURN STATUS:
                                0F42  4116 ;
                                0F42  4117 ;       Error status in R0.
                                0F42  4118 ;
                                0F42  4119 ;--
                                0F42  4120
                                0F42  4121 MAIN_ERR:
                          02    0F42  4122          BISL    #MFSM_EXIT_FLAG,-       ; Indicate exit in progress.
          00000476'EF           0F44  4123                  MISC_FLAGS              ;
    00002247'EF    05    E1    0F49  4124          BBC     #DVT$V_DISK,OUT_DEV,-   ; Branch if I/O device is a terminal.
                          03    0F50  4125                  5$                      ;
                        0094    31   0F51  4126          BRW     MAIN_EXIT           ; Otherwise, skip the following.
    000004A1'EF    50    D0    0F54  4127 5$:      MOVL    R0,ERR_MSGVEC+4         ; Save status for error message.
                                0F5B  4128
                                0F5B  4129          $QIOW_S -                       ; Make sure graphics mode is off.
                                0F5B  4130                  CHAN=OUT_CHAN,-
                                0F5B  4131                  FUNC=#IO$_WRITEVBLK,-
                                0F5B  4132                  P1=#^A//,-
                                0F5B  4133                  P2=#1
                                0F80  4134          JLBC    R0,MAIN_EXIT            ; Branch on error.
                                0F86  4135
0000054F'EF   00002216'EF   3C  0F86  4136          MOVZWL  SAVE_CHAR_BUF+2,VALUE   ; Get the original screen width.
        00000FBA'EF    00   FB  0F91  4137          CALLS   #0,SET_SCREEN           ; Reset the screen width.
                                0F98  4138          JLBC    R0,MAIN_EXIT            ; Branch on error.
                                0F9E  4139
                      0A2D  30  0F9E  4140          BSBW    SET_REGION_FULL         ; Reset the scrolling region to use the
                                0FA1  4141                                          ;  full screen. (Added for foreign
                                0FA1  4142                                          ;  terminals emulating VT100s).
```

```
                     0FA1  4143
  13 000004A1'EF  E9 0FA1  4144          BLSC    ERR_MSGVEC+4,10$        ; Don't clear the screen if exiting
                     0FA8  4145                                          ;  with error status.
00000476'EF  04  E1  0FA8  4146          BBC     #MF$V_CONT,MISC_FLAGS,- ; Don't clear the screen if display
                 0B      0FAF  4147                  10$                 ;  is not in continuous update mode.
              141D  30  0FB0  4148          BSBW    CLEAR_SCREEN            ; Clear the screen.
                     0FB3  4149          JLBC    R0,MAIN_EXIT            ; Branch on error.
                 00  11  0FB9  4150          BRB     10$                     ; Exit status is normal - don't output
                     0FBB  4151                                          ;  any message.
                     0FBB  4152
                 07  E1  0FBB  4153 10$:    BBC     #MF$V_NEW_CHAR,-        ; Branch if the terminal characteristics
  25 00000476'EF         0FBD  4154                  MISC_FLAGS,MAIN_EXIT    ;  were never changed.
                     0FC3  4155          $QIOW_S -                       ; Restore old terminal characteristics.
                     0FC3  4156                  CHAN=OUT_CHAN,-         ;
                     0FC3  4157                  FUNC=#IO$_SETMODE,-     ;
                     0FC3  4158                  P1=SAVE_CHAR_BUF,-      ;
                     0FC3  4159                  P2=#CHAR_BUF_LEN        ;
                     0FE8  4160
                     0FE8  4161 MAIN_EXIT:
000004A1'EF  01  D1  0FE8  4162          CMPL    #SS$_NORMAL,ERR_MSGVEC+4; Is the original exit status success?
                 07  13  0FEF  4163          BEQL    99$                     ; If so exit with status from MAIN_ERR
                     0FF1  4164                                          ;  routine.
50  000004A1'EF  D0  0FF1  4165          MOVL    ERR_MSGVEC+4,R0         ; Otherwise get the original status
                     0FF8  4166                                          ;  back.
                 04  0FF8  4167 99$:     RET                             ; And exit...
                     0FF9  4168
                     0FF9  4169
```

```
                     0FF9  4171 ;++
                     0FF9  4172 ;
                     0FF9  4173 ; FUNCTIONAL DESCRIPTION:
                     0FF9  4174 ;
                     0FF9  4175 ;       This routine coordinates the generation of each report by calling
                     0FF9  4176 ;       the appropriate routines based on the classes selected by the
                     0FF9  4177 ;       user.
                     0FF9  4178 ;
                     0FF9  4179 ;--
                     0FF9  4180
                     0FF9  4181 DISPLAY:
            0480 8F  BB  0FF9  4182          PUSHR   #^M<R7,R10>
 00002248'EF   01    D0  0FFD  4183          MOVL    #1,LINE_NUM                     ; Initialize line number.
1F 00002247'EF   01    E1  1004  4184          BBC     #DVT$V_DCRT,OUT_DEV,10$ ; Branch if output device isn't a
                           100C  4185                                                  ; DECcrt.
      0000224C'EF    DF  100C  4186          PUSHAL  COL_NUM                         ; Make sure line drawing and special
      00002248'EF    DF  1012  4187          PUSHAL  LINE_NUM                        ; graphics character set in G1 area in
      00000168'EF    DF  1018  4188          PUSHAL  GRAPHICS_DSCNTR                 ; DECcrt on each screen update.
 00000000'GF   03    FB  101E  4189          CALLS   #3,G^LIB$PUT_SCREEN             ;
                     1025  4190          JLBC    R0,DISPLAY_ERROR                ; Branch on error.
                     102B  4191
            08   8A  102B  4192 10$:     BICB    #SCR$M_MORE_DWN,-               ; Clear flag that indicates there is
      0000047A'EF       102D  4193                  SCROLL_FLAGS                    ; data off the bottom of the screen.
00000438'EF 00000434'EF  D0  1032  4194          MOVL    LINE_DIF,PRIOR_LINE_DIF ; Save record of lines changed on the
                           103D  4195                                                  ; last pass.
00000434'EF 00000444'EF  D0  103D  4196          MOVL    SCROLL_DIF,LINE_DIF     ; Clear record of lines changed this
                           1048  4197                                                  ; pass, but set the bits for any
                           1048  4198                                                  ; lines that have to be redrawn
                           1048  4199                                                  ; because of scrolling.
      0000043C'EF    D0  1048  4200          MOVL    BLANK_LINES,-           ; Save record of blank lines on the
      00000440'EF       104E  4201                  PRIOR_BLANK_LINES               ; last pass.
      0000043C'EF    D4  1053  4202          CLRL    BLANK_LINES             ; Clear record of blank lines this
                           1059  4203                                                  ; pass.
18 00000476'EF   03    E1  1059  4204          BBC     #MF$V_RIP,MISC_FLAGS,20$; Branch if refresh not in progress.
      000021C8'EF    D4  1061  4205          CLRL    CLUB_HEADER_SIZE        ; Initialize CLUB header size...
      000021CC'EF    D4  1067  4206          CLRL    SCS_HEADER_SIZE         ; ...SCS header size...
      00000427'EF    94  106D  4207          CLRB    CLUB_NUM_FIELDS         ; ...number of CLUB fields...
      00000426'EF    94  1073  4208          CLRB    SCS_NUM_FIELDS          ; ...number of SCS fields...
                     1079  4209
                     1079  4210 ;
                     1079  4211 ; Go allocate some buffer space, lock in memory and take a snapshot of the
                     1079  4212 ; SCS and port databases.
                     1079  4213 ;
                     1079  4214
          F4B1 CF    16  1079  4215 20$:     JSB     DO_SNAPSHOT             ;
                     107D  4216          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
                     1083  4217          JBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
                     1083  4218                  CURRENT_REPORT,-                ; cluster report.
                     1083  4219                  DISPLAY_PTR             ;
      0000007F 8F    D3  108E  4220          BITL    #CLS$M_ANY_CLUS,-       ; Any Cluster report information to be
      00000001'EF       1094  4221                  CLASS_SELECT                    ; displayed?
            03   12  1099  4222          BNEQ    30$                     ; Branch if so.
          01A7   31  109B  4223          BRW     DISPLAY_END             ; And go finish up.
            00   E1  109E  4224 30$:     BBC     #CLS$V_CLUB,-           ; Branch if display of CLUB is disabled.
    63 00000001'EF       10A0  4225                  CLASS_SELECT,DISPLAY_SCS;
                     10A6  4226 ;
                     10A6  4227 ; Display the CLUSTER class at the top of the screen.  This class
```

```
                                10A6  4228 ; contains general cluster information from the CLUB.
                                10A6  4229 ;
                                10A6  4230
                                10A6  4231 DISPLAY_CLUB:
          00000001'EF    D0     10A6  4232          MOVL    CLASS_SELECT,-          ; Save mask of selected classes.
          000021FC'EF           10AC  4233          SAVED_CLASSES           ;
          0000007E 8F    CA     10B1  4234          BICL    #CLS$M_ANY_SCS,-        ; Temporarily disable all classes but
          00000001'EF           10B7  4235          CLASS_SELECT            ;  the CLUB.
     000021F8'EF   01    C8     10BC  4236          BISL    #CLS$M_CLUB,DATA_CNTRL  ; Enable display of fields in the CLUB.
                   01    D0     10C3  4237          MOVL    #CLS$M_CLUB,-           ; Set current class to CLUSTER.
          000021F4'EF           10C5  4238          CURRENT_CLASS           ;
                0B79    30     10CA  4239          BSBW    GET_HEADER              ; Output CLUB header to screen buffer.
                               10CD  4240          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
          000021C8'EF    D0     10D3  4241          MOVL    CLUB_HEADER_SIZE,-      ; If no refresh and displaying both CLUB
          000021C4'EF           10D9  4242          HEADER_SIZE             ;  and SCS, the HEADER_SIZE will be set
                                10DE  4243                                  ;  to the SCS header size, so we must
                                10DE  4244                                  ;  get the real CLUB_HEADER_SIZE back.
          0000004B'EF    DF     10DE  4245          PUSHAL  CLUB_BUF                ; Push address of the CLUB.
     00002729'EF   01    FB     10E4  4246          CALLS   #1,GET_DATA             ; Get a line of data.
                               10EB  4247 ;
                               10EB  4248 ; Since the number of lines in the CLUB portion of the display will never
                               10EB  4249 ; change, we don't have to call DISPLAY_COMMON.  Instead we output the
                               10EB  4250 ; bottom line of the display box only when doing a refresh pass.
                               10EB  4251 ;
05 00000476'EF   03    E1     10EB  4252          BBC     #MF$V_RIP,MISC_FLAGS,20$; Branch if refresh not in progress.
                0FFB    30     10F3  4253          BSBW    DRAW_LAST_LINE          ; Draw bottom line of CLUB display box.
                  06    11     10F6  4254          BRB     30$                     ; Branch to common code.
          00002248'EF    D6     10F8  4255 20$:     INCL    LINE_NUM                ; Skip over bottom line of CLUB display
                               10FE  4256                                  ;  box.
          000021FC'EF    D0     10FE  4257 30$:     MOVL    SAVED_CLASSES,-         ; Restore mask of selected classes.
          00000001'EF           1104  4258          CLASS_SELECT            ;
                               1109  4259
                               1109  4260 ;
                               1109  4261 ; Display any SCS data (SYSTEMS, CIRCUITS, CONNECTION, COUNTERS,
                               1109  4262 ; or CREDITS classes) or any CSB data (MEMBERS class).
                               1109  4263 ;
                               1109  4264
                               1109  4265 DISPLAY_SCS:
          0000007E 8F    D3     1109  4266          BITL    #CLS$M_ANY_SCS,-        ; Any SCS or Cluster System Block info
          00000001'EF           110F  4267          CLASS_SELECT            ;  to be displayed?
                               1114  4268          JEQL    DISPLAY_END             ; Branch if not.
          00000001'EF    D0     1119  4269          MOVL    CLASS_SELECT,-          ; Save mask of selected classes.
          000021FC'EF           111F  4270          SAVED_CLASSES           ;
00000460'EF 0000045C'EF   D0     1124  4271          MOVL    V_SCROLL,V_SCROLL_LINE  ; Set the number of data lines to skip.
                F7C9'   30     112F  4272          BSBW    GET_NUM_PB              ; Get the number of circuits on each
                               1132  4273                                  ;  system.
          00000001'EF   01    CA     1132  4274          BICL    #CLS$M_CLUB,CLASS_SELECT; We've already displayed CLUB info.
          000021F4'EF   02    D0     1139  4275          MOVL    #CLS$M_SYS,CURRENT_CLASS; Set current class to SYSTEMS.
                0B03    30     1140  4276          BSBW    GET_HEADER              ; Output SCS header to screen buffer.
                               1143  4277          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
          000021CC'EF    D0     1149  4278          MOVL    SCS_HEADER_SIZE,-       ; If no refresh, and displaying both
          000021C4'EF           114F  4279          HEADER_SIZE             ;  CLUB and SCS, the header size will
                               1154  4280                                  ;  be set to the CLUB header size.  We
                               1154  4281                                  ;  must get the SCS header size back.
                  0C    13     1154  4282          BEQL    10$                     ; Branch if nothing to display.
                015C    30     1156  4283          BSBW    SCS_REPORT              ; Output the SCS data.
                00FE    30     1159  4284          BSBW    DISPLAY_COMMON          ; Execute common display code.
```

```
                     115C  4285          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
000021FC'EF   D0     1162  4286 10$:     MOVL    SAVED_CLASSES,-         ; Restore mask of selected classes.
00000001'EF          1168  4287          CLASS_SELECT            ;
        00D5   31    116D  4288          BRW     DISPLAY_END             ; Finish up display.
                     1170  4289
                     1170  4290
                     1170  4291 ;
                     1170  4292 ; Display the Port-to-Port data.  (LOCAL_PORTS and ERRORS classes).
                     1170  4293 ;
                     1170  4294
                     1170  4295 DISPLAY_PTP:
00000460'EF   D4     1170  4296          CLRL    V_SCROLL_LINE           ; Insure that vertical scrolling
                     1176  4297                                          ;  doesn't occur in the local report.
00000080 8F   D0     1176  4298          MOVL    #CLS$M_LPRT,-           ; Set current class to LOCAL_PORTS.
000021F4'EF          117C  4299          CURRENT_CLASS           ;
        0AC2   30    1181  4300          BSBW    GET_HEADER              ; Output SCS header to screen buffer.
                     1184  4301          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
        0386   30    118A  4302          BSBW    LOCAL_REPORT            ; Output the PTP data.
                     118D  4303          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
        00C4   30    1193  4304          BSBW    DISPLAY_COMMON          ; Execute common display code.
                     1196  4305          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
                     119C  4306 ;
                     119C  4307 ; For the Local_Ports report, we also output the number of buffer descriptors
                     119C  4308 ; free and in use below the rest of the display.
                     119C  4309 ;
        125E   30    119C  4310          BSBW    ERASE_TWO_LINES         ; Leave a couple of blank lines.
                     119F  4311          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
   5A   00000023'EF  DE    11A5  4312          MOVAL   NUM_BUF_DSCR_DSCR,R10   ; Point to descriptor for the number of
                     11AC  4313                                          ; buffer descriptors in use.
    57   10 AA   D0    11AC  4314          MOVL    FD$L_OFFSET(R10),R7     ; Get the SCS offset from the
                     11B0  4315                                          ;  descriptor.
         13CC    30    11B0  4316          BSBW    FORMAT_DATA             ; Format the data.
                     11B3  4317          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
     0000224C'EF   3F    11B9  4318          PUSHAW  COL_NUM                 ; Output the data to the report.
     00002248'EF   3F    11BF  4319          PUSHAW  LINE_NUM                ;
     00002153'EF   DF    11C5  4320          PUSHAL  FORMAT_BUF_DSCR         ;
00000000'GF   03    FB    11CB  4321          CALLS   #3,G^LIB$PUT_SCREEN     ;
                     11D2  4322          JLBC    R0,DISPLAY_ERROR        ; Branch on error.
     000001DF'EF   DF    11D8  4323          PUSHAL  BUFF_MSG                ; Print "CI buffer descriptors in use."
00000000'GF   01    FB    11DE  4324          CALLS   #1,G^LIB$PUT_SCREEN     ;
           6D 50    E9    11E5  4325          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
00000000'GF   00    FB    11E8  4326          CALLS   #0,G^LIB$ERASE_LINE     ; Erase the rest of the line.
           63 50    E9    11EF  4327          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
     00002248'EF   D6    11F2  4328          INCL    LINE_NUM                ; Free buffer descriptors on next line.
     0000004C'EF   DE    11F8  4329          MOVAL   NUM_FREE_BUF_DSCR_DSCR,-; Point to descriptor for number of
                5A         11FE  4330          R10                     ;  buffer descriptors free.
    57   10 AA   D0    11FF  4331          MOVL    FD$L_OFFSET(R10),R7     ; Get the SCS offset from the
                     1203  4332                                          ;  descriptor.
         1379    30    1203  4333          BSBW    FORMAT_DATA             ; Format the data.
           4C 50    E9    1206  4334          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
     0000224C'EF   3F    1209  4335          PUSHAW  COL_NUM                 ; Output the data to the report.
     00002248'EF   3F    120F  4336          PUSHAW  LINE_NUM                ;
     00002153'EF   DF    1215  4337          PUSHAL  FORMAT_BUF_DSCR         ;
00000000'GF   03    FB    121B  4338          CALLS   #3,G^LIB$PUT_SCREEN     ;
           30 50    E9    1222  4339          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
     00000205'EF   DF    1225  4340          PUSHAL  FREE_BUFF_MSG           ; Print "CI buffer descriptors free."
00000000'GF   01    FB    122B  4341          CALLS   #1,G^LIB$PUT_SCREEN     ;
```

```
                     20 50   E9  1232  4342          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
00000000'GF          00      FB  1235  4343          CALLS   #0,G^LIB$ERASE_LINE     ; Erase the rest of the line.
                     16 50   E9  123C  4344          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
          00002248'EF        D6  123F  4345          INCL    LINE_NUM                ; Set line number to next line.
                             1245  4346
                             1245  4347 DISPLAY_END:
                             1245  4348
                             1245  4349 ;
                             1245  4350 ; Erase any lines following the report, and update the scrolling
                             1245  4351 ; signals and broadcast messages at the bottom of the display.
                             1245  4352 ;
                             1245  4353
                       0851  30  1245  4354          BSBW    ERASE_PAGE              ; Erase any remaining data.
                     0A 50   E9  1248  4355          BLBC    R0,DISPLAY_ERROR        ; Branch on error.
                       0686  30  124B  4356          BSBW    CHECK_MORE              ; Add scrolling signals or broadcast
                             124E  4357                                            ;  messages to the report.
00000476'EF          08      CA  124E  4358          BICL    #MF$M_RIP,MISC_FLAGS    ; Clear refresh in progress flag.
                             1255  4359
                             1255  4360 DISPLAY_ERROR:
                     0480 8F     BA  1255  4361          POPR    #^M<R7,R10>
                             05  1259  4362          RSB
                             125A  4363
                             125A  4364
```

```
                                    125A  4366 ;
                                    125A  4367 ; This routine is called output of either the Cluster Report or the Local
                                    125A  4368 ; report. It determines if the bottom line of the box surrounding the data
                                    125A  4369 ; should be draw.  This line is drawn only if there is room on the screen and
                                    125A  4370 ; either the line won't be drawn in the same place is was last pass, or the
                                    125A  4371 ; screen is being refreshed.
                                    125A  4372 ;
                                    125A  4373
                                    125A  4374 DISPLAY_COMMON:
                         50   01    D0  125A  4375          MOVL    #SS$_NORMAL,R0          ; Assume success.
00000428'EF      00002248'EF       D0  125D  4376          MOVL    LINE_NUM,NUM_LINES      ; Save number of lines this pass.
                 00000444'EF       D4  1268  4377          CLRL    SCROLL_DIF              ; Clear record of lines to redraw
                                    126E  4378                                          ;  because we scrolled.
                         03   E0    126E  4379          BBS     #SCR$V_MORE_DWN,-       ; Branch if we have more lines of data
                    3E 0000047A'EF      1270  4380                  SCROLL_FLAGS,20$        ;  than will fit on the screen.
0000225C'EF      00002248'EF       D1  1276  4381          CMPL    LINE_NUM,CMD_LINE       ; Will the next line be in the command
                                    1281  4382                                          ;  region?
                         31   13    1281  4383          BEQL    20$                     ; If so don't output the last line.
                         08   E4    1283  4384          BBSC    #MF$V_SCROLL,-          ; Clear flag that indicates that we
                    20 00000476'EF      1285  4385                  MISC_FLAGS,10$          ;  have scrolled.
                 00000428'EF       D1  128B  4386          CMPL    NUM_LINES,-             ; Do we have the same number of lines
                 0000042C'EF         1291  4387                  PRIOR_NUM_LINES         ;  of data as last pass?
                         13   12    1296  4388          BNEQ    10$                     ; If not, draw the last line of the box.
                 0B 00000476'EF   03   E0  1298  4389          BBS     #MF$V_RIP,MISC_FLAGS,10$; If refresh in progress better draw the
                                    12A0  4390                                          ;  last line.
                 00002248'EF       D6  12A0  4391          INCL    LINE_NUM                ; Skip the last line draw.
                         50   01    D0  12A6  4392          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                         09   11    12A9  4393          BRB     20$                     ; Jump to common code.
                        0E43  30    12AB  4394 10$:     BSBW    DRAW_LAST_LINE          ; Draw bottom line of display box.
                                    12AE  4395          JLBC    R0,MAIN_ERR             ; Branch on error.
                         05   12B4  4396 20$:     RSB
                                    12B5  4397
```

```
12B5  4399          .SBTTL  SCS report generation routine.
12B5  4400 ;++
12B5  4401 ;
12B5  4402 ; FUNCTIONAL DESCRIPTION:
12B5  4403 ;
12B5  4404 ;       This routine formats the SCS portion of the cluster report.  This
12B5  4405 ;       portion of the report has been designed to illustrate the heirarchy of
12B5  4406 ;       the cluster by showing the relationship between a system on the cluster,
12B5  4407 ;       the circuits established on the system, and the connections established
12B5  4408 ;       over these circuits.  This portion of the report also displays the
12B5  4409 ;       system membership status (MEMBERS class).
12B5  4410 ;
12B5  4411 ;       This routine scans through the snapshot buffer looking for data to be
12B5  4412 ;       displayed while controlling what data is to be displayed on a
12B5  4413 ;       particular line of the report (using the mask at DATA_CNTRL).
12B5  4414 ;       If the data found in the snapshot buffer conforms to the data to be
12B5  4415 ;       displayed on the particular line and the user has elected to see that
12B5  4416 ;       type of data, then a call is made to the GET_DATA routine to extract,
12B5  4417 ;        format and add the line of data to the screen buffer.
12B5  4418 ;
12B5  4419 ;       This routine also sets up the counter V_SCROLL_LINE by using the
12B5  4420 ;       value in V_SCROLL.  This count is an input to the GET_DATA routine
12B5  4421 ;       which causes it to ignore the first "V_SCROLL" times it is called.
12B5  4422 ;       This effectively scrolls the report up "V_SCROLL" lines.
12B5  4423 ;
12B5  4424 ; CALLING SEQUENCE:
12B5  4425 ;
12B5  4426 ;       JSB     SCS_REPORT
12B5  4427 ;
12B5  4428 ; CALLED BY:
12B5  4429 ;
12B5  4430 ;       DISPLAY_SCS
12B5  4431 ;
12B5  4432 ; INPUTS:
12B5  4433 ;
12B5  4434 ;       V_SCROLL        Set to the number of lines of data to skip before
12B5  4435 ;                       adding the first line of data to the report.
12B5  4436 ;
12B5  4437 ;       CLASS_SELECT    Contains a mask of the classes selected for display
12B5  4438 ;                       by the user.
12B5  4439 ;
12B5  4440 ; IMPLICIT INPUTS:
12B5  4441 ;
12B5  4442 ;       Snapshot has been performed and all the system blocks, path blocks, etc.
12B5  4443 ;       have been stored in SNAP_BUF.
12B5  4444 ;
12B5  4445 ; IMPLICIT OUTPUTS:
12B5  4446 ;
12B5  4447 ;       Display data stored in SCREEN_BUF.
12B5  4448 ;
12B5  4449 ; RETURN STATUS:
12B5  4450 ;
12B5  4451 ;       Status from call to GET_DATA routine passed in R0.
12B5  4452 ;
12B5  4453 ; REGISTER USAGE:
12B5  4454 ;
12B5  4455 ;       R8  -   Pointer to connections in CDT buffer.
```

```
                        12B5  4456 ;       R9  -   Pointer to path blocks in PB buffer.
                        12B5  4457 ;       R10 -   Pointer to cluster system block in CSB buffer.
                        12B5  4458 ;       R11 -   Pointer to system blocks in SB buffer.
                        12B5  4459 ;
                        12B5  4460 ;
                        12B5  4461 ; NOTES:
                        12B5  4462 ;       In order to display a class, the corresponding CLS$ (class) bit in
                        12B5  4463 ;       CLASS_SELECT must be set.  The same must be true of DATA_CNTRL or all
                        12B5  4464 ;       the fields in that class will be filled in with blanks.  Normally the
                        12B5  4465 ;       CLASS_SELECT bits are controlled by the user through the ADD class and
                        12B5  4466 ;       REMOVE class commands and the DATA_CNTRL bits are controlled by Show
                        12B5  4467 ;       Cluster to format the display.  When the CLUSTER class is being
                        12B5  4468 ;       displayed, Show Cluster also manipulates the CLASS_SELECT bits.
                        12B5  4469 ;       CLASS_SELECT bits are manipulated in this routine and in the REFRESH
                        12B5  4470 ;       routine to control the display of the header.
                        12B5  4471 ;--
                        12B5  4472
                        12B5  4473 ASSUME  CLS$V_750_SYS EQ 16
                        12B5  4474
                        12B5  4475 SCS_REPORT:
                0F04 8F   BB  12B5  4476          PUSHR   #^M<R2,R8,R9,R10,R11>
                          CA  12B9  4477          BICL    #<CLS$M_CSB!CLS$M_CIR!- ; Assume no members, circuits,
                              12BA  4478                    CLS$M_CON!CLS$M_CNT!- ;  connections, counters, or credits,
                              12BA  4479                    CLS$M_CRD!CLS$M_CLUB>,-; and no general cluster information
000021F8'EF   0000007D 8F     12BA  4480                    DATA_CNTRL             ;  to display, so disable them all.
          5B  000003ED'EF   D0  12C4  4481          MOVL    VAR_BUF,R11            ; Get address of first system block.
          52  000004B2'EF   DE  12CB  4482 10$:     MOVAL   HW_TYPE_TABLE,R2       ; Point to table of system types.
                        51  D4  12D2  4483          CLRL    R1                     ; Initialize index into table.
              34 AB   6241  D1  12D4  4484 20$:     CMPL    (R2)[R1],-             ; Is the type of system specified by
                              12D9  4485                    SB$T_HWTYPE(R11)       ;  this system block in the table?
                        06  13  12D9  4486          BEQL    30$                    ; Branch if so.
                 F5 51  06  F2  12DB  4487          AOBLSS  #NUM_HW_TYPES,R1,20$   ; Loop for all types of systems.
                              12DF  4488 ;
                              12DF  4489 ; If not in the table, accept it anyway.  It could be the local system block
                              12DF  4490 ; and VMSVAXCLUSTER <> 1, or pre VMS Version 3.3 system.
                              12DF  4491 ;
                        0E  11  12DF  4492          BRB     40$                    ;
                              12E1  4493
                     51 10  C0  12E1  4494 30$:     ADDL    #CLS$V_750_SYS,R1      ; Convert index to CLS$ bit value.
                              12E4  4495          JBC     R1,CLASS_SELECT,160$   ; Branch if user has removed this
                              12EF  4496                                         ;  type of system from the display.
          000014F8'EF     16  12EF  4497 40$:     JSB     CHECK_ID               ; Check if this systems ID is in the
                              12F5  4498                                         ;  disabled list.
                              12F5  4499          JLBS    R0,160$                ; Go on to next system block if it is.
000021F8'EF   02  C8  12FB  4500          BISL    #CLS$M_SYS,DATA_CNTRL  ; Enable display of systems class.
     00002248'E   90  1302  4501          MOVB    LINE_NUM,-             ; Store line number for this system in
           09 AB      1308  4502                  SHC$SB$B_DIS_POS(R11)  ;  system block.
              02  E1  130A  4503 50$:     BBC     #CLS$V_CSB,-           ; Branch if user has removed members
  14 00000001'EF      130C  4504                  CLASS_SELECT,55$       ;  from the display.
000021F8'EF   04  CA  1312  4505          BICL    #CLS$M_CSB,DATA_CNTRL  ; Assume no CSB associated with this
                    1319  4506                                         ;  system.
       5A   5C AB  D0  1319  4507          MOVL    SB$L_CSB(R11),R10      ; Get address of CSB for this system.
              07  13  131D  4508          BEQL    55$                    ; Branch if no CSB.
000021F8'EF   04  C8  131F  4509          BISL    #CLS$M_CSB,DATA_CNTRL  ; Enable display of CSB's.
     0000144C'EF  16  1326  4510 55$:     JSB     CHECK_NUM_PB           ; See if the number of path blocks on
                    132C  4511                                         ;  this system block has changed.
       59   0C AB  D0  132C  4512          MOVL    SB$L_PBFL(R11),R9      ; Get address of first path block on
```

```
                          1330  4513                                  ;  this system block.
                 6B   13  1330  4514          BEQL    120$            ; Branch if there are no circuits on
                          1332  4515                                  ;  this system.
              12 A9   B1  1332  4516          CMPW    PB$W_STATE(R9),-        ; Is the circuit in the open state?
                 03       1335  4517          #PB$C_OPEN              ;
                 0A   12  1336  4518          BNEQ    70$             ; Branch if not.
                 09   E1  1338  4519 60$:     BBC     #CLS$V_OPEN_CIR,-       ; Branch if user has removed open
     11 00000001'EF       133A  4520          CLASS_SELECT,90$        ;  circuits from the display.
                 08   11  1340  4521          BRB     80$
                 0A   E1  1342  4522 70$:     BBC     #CLS$V_NOOPEN_CIR,-     ; Branch if user has removed circuits in
     07 00000001'EF       1344  4523          CLASS_SELECT,90$        ;  states other that open from the
                          134A  4524                                  ;  display.
 000021F8'EF     08   C8  134A  4525 80$:     BISL    #CLS$M_CIR,DATA_CNTRL   ; Enable display of circuits class.
      0000147A'EF    16   1351  4526 90$:     JSB     CHECK_NUM_CDT           ; See if the number of connections
                          1357  4527                                  ;  on this path has changed since the
                          1357  4528                                  ;  last pass.
        58   34 A9   D0  1357  4529          MOVL    PB$L_CDTLST(R9),R8      ; Get address of first CDT on this
                          135B  4530                                  ;  path block.
                 40   13  135B  4531          BEQL    120$            ; Branch if there are no connections
                          135D  4532                                  ;  on this circuit.
              18 A8   B1  135D  4533          CMPW    SHC$L_CDT_DIF+CDT$W_STATE(R8),-
                 01       1360  4534          #CDT$C_LISTEN           ; Is this CDT in the "listen" state?
                 10   12  1361  4535          BNEQ    110$            ; Branch if not.
 000021F8'EF     08   CA  1363  4536          BICL    #CLS$M_CIR,DATA_CNTRL   ; Otherwise, disable display of
                          136A  4537                                  ;  circuits class.
 000021F8'EF     10   C8  136A  4538          BISL    #CLS$M_CON,DATA_CNTRL   ; Enable display of connections class.
                 2A   11  1371  4539          BRB     120$            ; Jump to common code.
              18 A8   B1  1373  4540 110$:     CMPW    SHC$L_CDT_DIF+CDT$W_STATE(R8),-
                 02       1376  4541          #CDT$C_OPEN             ; Is this CDT in the "open" state?
                 0A   12  1377  4542          BNEQ    115$            ; Branch if not.
                 0B   E1  1379  4543          BBC     #CLS$V_OPEN_CON,-       ; Branch if user has removed open
     11 00000001'EF       137B  4544          CLASS_SELECT,119$       ;  open connections from the display.
                 08   11  1381  4545          BRB     118$            ; Branch to common.
                 0C   E1  1383  4546 115$:     BBC     #CLS$V_NOOPEN_CON,-     ; Branch if user has removed non-open
       07 00000001'EF     1385  4547          CLASS_SELECT,119$       ;  connections from the display.
      000021F8'EF    10   C8  138B  4548 118$:     BISL    #CLS$M_CON,DATA_CNTRL   ; Enable display of connections class.
          00000060 8F   C8  1392  4549 119$:     BISL    #CLS$M_CRD!CLS$M_CNT,-  ; Enable display of counters and
          000021F8'EF       1398  4550          DATA_CNTRL              ;  credits classes.
                          139D  4551
                          139D  4552 ;
                          139D  4553 ; AND the mask of classes we've enabled with the mask of classes the user has
                          139D  4554 ; selected and see if there's any data to be displayed.  If so, output a line
                          139D  4555 ; of data, otherwise continue looking.
                          139D  4556 ;
                          139D  4557
      51   000021F8'EF    D2  139D  4558 120$:     MCOML   DATA_CNTRL,R1           ; Get mask of classes enabled.
51    00000001'EF    51   CB  13A4  4559          BICL3   R1,CLASS_SELECT,R1      ; Any data to be displayed?
                 2A   13  13AC  4560          BEQL    130$            ; Branch if not.
      00000476'EF    03   E0  13AE  4561          BBS     #MF$V_RIP,MISC_FLAGS,-  ; Skip position check if performing a
                 0E       13B5  4562          125$                    ;  refresh.
                 08   E0  13B6  4563          BBS     #MF$V_SCROLL,-          ; Skip position check if we just
        06 00000476'EF     13B8  4564          MISC_FLAGS,125$         ;  scrolled.
          000014A8'EF    16   13BE  4565          JSB     CHECK_POS               ; Check if the system has moved in the
                          13C4  4566                                  ;  display.
                 58   DD  13C4  4567 125$:     PUSHL   R8              ; Push address of CDT.
                 59   DD  13C6  4568          PUSHL   R9              ; Push address of path block.
                 5A   DD  13C8  4569          PUSHL   R10             ; Push address of cluster system block.
```

```
                     5B   DD  13CA  4570          PUSHL   R11                             ; Push address of system block.
                     00   DD  13CC  4571          PUSHL   #0                              ; Bogus CLUB address.
          00002729'EF     05  FB  13CE  4572          CALLS   #5,GET_DATA                     ; Get a line of data.
                   6F    50  E9  13D5  4573          BLBC    R0,999$                         ; Branch on error.
000021F8'EF  00000070 8F   D3  13D8  4574 130$:    BITL    #<CLS$M_CON!CLS$M_CNT!-         ; Are we displaying connections,
                               13E3  4575                    CLS$M_CRD>,DATA_CNTRL           ;  counters or credits?
                     0C    13  13E3  4576          BEQLU   140$                            ; Branch if not.
          000C21F8'EF     0E  CA  13E5  4577          BICL    #<CLS$M_SYS!CLS$M_CSB!-         ; Disable display of systems, members,
                               13EC  4578                    CLS$M_CIR>,DATA_CNTRL           ;  and circuits class.
                  58   68    D0  13EC  4579          MOVL    (R8),R8                         ; Get address of next CDT.
                     AC    12  13EF  4580          BNEQ    120$                            ; Loop for all connections on this
                               13F1  4581                                                    ;  circuit.
000021F8'EF  00000070 8F   CA  13F1  4582 140$:    BICL    #<CLS$M_CON!CLS$M_CNT!-         ; Assume no connections on the next
                               13FC  4583                    CLS$M_CRD>,DATA_CNTRL           ;  circuit, so disable display of
                               13FC  4584                                                    ;  connections, counters, and credits
                               13FC  4585                                                    ;  classes.
                     59    D5  13FC  4586          TSTL    R9                              ; Are there any circuits on this system?
                     1A    13  13FE  4587          BEQL    160$                            ; Branch if not.
                  59   69    D0  1400  4588          MOVL    (R9),R9                         ; Get address of next path block.
                     0A    13  1403  4589          BEQL    150$                            ; Branch if no more circuits on this
                               1405  4590                                                    ;  system.
                     06    CA  1405  4591          BICL    #CLS$M_SYS!CLS$M_CSB,-          ; Disable display of systems, members
          000021F8'EF             1407  4592                    DATA_CNTRL                      ;  classes.
                   FF29    31  140C  4593          BRW     60$                             ; Loop for all circuits on this system.
                     CA    140F  4594 150$:    BICL    #<CLS$M_CIR!CLS$M_CON!-         ; Assume no circuits on next system, so
                               1410  4595                    CLS$M_CNT!CLS$M_CRD>,-          ;  disable display of circuits,
000021F8'EF  00000078 8F         1410  4596                    DATA_CNTRL                      ;  connections, counters, and credits.
          51    000001F3'EF     DE  141A  4597 160$:    MOVAL   DUMMY_SB,R1                     ; Determine if we are displaying
                  5B   51    D1  1421  4598          CMPL    R1,R11                          ;  formative paths without formative
                               1424  4599                                                    ;  system blocks.
                     21    13  1424  4600          BEQL    999$                            ; If so, end the report.
                  5B   6B    D0  1426  4601          MOVL    (R11),R11                       ; Get address of next system block.
                               1429  4602          JNEQ    10$                             ; Loop for all systems in the cluster.
                               142E  4603 ;
                               142E  4604 ; Display any formative path blocks that have no associated formative system
                               142E  4605 ; block.
                               142E  4606 ;
5B   000001F3'EF     DE  142E  4607          MOVAL   DUMMY_SB,R11                    ; Get the address of the fake system
                               1435  4608                                                    ;  block.
          0C AB      D5  1435  4609          TSTL    SB$L_PBFL(R11)                  ; Are there any formative path blocks
                               1438  4610                                                    ;  on the dummy system block?
                     0D    13  1438  4611          BEQL    999$                            ; Branch if not.
                   10E3    30  143A  4612          BSBW    GET_BLANK                       ; Leave one blank line.
                     06    CA  143D  4613          BICL    #CLS$M_SYS!CLS$M_CSB,-          ; Disable display of systems, and
          000021F8'EF             143F  4614                    DATA_CNTRL                      ;  members classes of data.
                   FEC3    31  1444  4615          BRW     50$                             ; Display the formative path blocks.
                               1447  4616
          0F04 8F    BA  1447  4617 999$:    POPR    #^M<R2,R8,R9,R10,R11>
                     05  144B  4618          RSB
                         144C  4619
                         144C  4620
```

```
                     144C  4622 ;++
                     144C  4623 ;
                     144C  4624 ; FUNCTIONAL DESCRIPTION:
                     144C  4625 ;
                     144C  4626 ;       This routine insures that a change in the number of path blocks
                     144C  4627 ;       does not go undetected.  When a path block is removed from the
                     144C  4628 ;       database, CHECK_DIF is no longer called to check for a change
                     144C  4629 ;       in data for that path block.  This means that the line will
                     144C  4630 ;       not be redrawn until the screen is refreshed.  This routine
                     144C  4631 ;       calls CHECK_DIF with the data to be checked being the number
                     144C  4632 ;       of path blocks, so a change will always be detected.
                     144C  4633 ;
                     144C  4634 ; CALLING SEQUENCE:
                     144C  4635 ;
                     144C  4636 ;       JSB     CHECK_NUM_PB
                     144C  4637 ;
                     144C  4638 ; CALLED BY:
                     144C  4639 ;
                     144C  4640 ;       CLUSTER_REPORT
                     144C  4641 ;
                     144C  4642 ; INPUTS:
                     144C  4643 ;
                     144C  4644 ;       Address of the system block in R11
                     144C  4645 ;
                     144C  4646 ; OUTPUTS:
                     144C  4647 ;
                     144C  4648 ;       The bit coresponding to the current line number in LINE_DIF
                     144C  4649 ;       is set if the number of path block on this system block has
                     144C  4650 ;       changed.
                     144C  4651 ;
                     144C  4652 ;--
                     144C  4653
                     144C  4654 CHECK_NUM_PB:
                0480 8F    BB  144C  4655          PUSHR   #^M<R7,R10>
             57   04 AB    DE  1450  4656          MOVAL   SB$SHC$L_NUM_PB(R11),R7 ; Get the address of the data.
           000001A9'EF    DE  1454  4657          MOVAL   SB$SHC$L_NUM_PB_DSCR,-  ; Get the address of the field
                   5A          145A  4658                  R10                     ;  descriptor.
           00002D3D'EF    16  145B  4659          JSB     CHECK_DIF               ; See if the number of path
                               1461  4660                                          ;  blocks changed.
                   02     E5  1461  4661          BBCC    #MF$V_INVERSE,-         ; Branch if there was no change.
                0C 00000476'EF 1463  4662                  MISC_FLAGS,99$          ;
00 00000434'EF  00002248'EF    E2  1469  4663          BBSS    LINE_NUM,LINE_DIF,99$   ; Indicate that this line of data
                               1475  4664                                          ;  must be redrawn.
                0480 8F    BA  1475  4665 99$:     POPR    #^M<R7,R10>
                           05  1479  4666          RSB
                               147A  4667
```

```
                        147A  4669 ;++
                        147A  4670 ;
                        147A  4671 ; FUNCTIONAL DESCRIPTION:
                        147A  4672 ;
                        147A  4673 ;       This routine insures that a change in the number of connections
                        147A  4674 ;       does not go undetected.  When a CDT is removed from the database
                        147A  4675 ;       CHECK_DIF is no longer called to check for a change in data for
                        147A  4676 ;       that CDT.  This means that the line will not be redrawn until
                        147A  4677 ;       screen is refreshed.  This routine calls CHECK_DIF with the data
                        147A  4678 ;       to be checked being the number of CDTS , so a change will always
                        147A  4679 ;       be detected.
                        147A  4680 ;
                        147A  4681 ; CALLING SEQUENCE:
                        147A  4682 ;
                        147A  4683 ;       JSB     CHECK_NUM_CDT
                        147A  4684 ;
                        147A  4685 ; CALLED BY:
                        147A  4686 ;
                        147A  4687 ;       CLUSTER_REPORT
                        147A  4688 ;
                        147A  4689 ; INPUTS:
                        147A  4690 ;
                        147A  4691 ;       Address of Path Block in R11
                        147A  4692 ;
                        147A  4693 ; OUTPUTS:
                        147A  4694 ;
                        147A  4695 ;       The bit coresponding to the current line number in LINE_DIF
                        147A  4696 ;       is set if the number of CDT's on this path block has changed.
                        147A  4697 ;
                        147A  4698 ;--
                        147A  4699
                        147A  4700 CHECK_NUM_CDT:
                0480 8F BB  147A  4701          PUSHR   #^M<R7,R10>
             57   04 AB DE  147E  4702          MOVAL   PB$SHC$L_NUM_CDT(R11),R7; Get the address of the data.
           00000501'EF DE  1482  4703          MOVAL   PB$SHC$L_NUM_CDT_DSCR,- ; Get the address of the field
                    5A     1488  4704                  R10                     ;  descriptor.
           00002D3D'EF 16  1489  4705          JSB     CHECK_DIF               ; See if the number of path
                           148F  4706                                          ;  blocks changed.
                    02 E5  148F  4707          BBCC    #MF$V_INVERSE,-         ; Branch if there was no change.
                0C 00000476'EF  1491  4708                  MISC_FLAGS,99$      ;
00 00000434'EF   00002248'EF E2  1497  4709          BBSS    LINE_NUM,LINE_DIF,99$   ; Indicate that this line of data
                           14A3  4710                                          ;  must be redrawn.
                0480 8F    BA  14A3  4711 99$:      POPR    #^M<R7,R10>
                           05  14A7  4712          RSB
                           14A8  4713
```

```
                     14A8  4715 ;++
                     14A8  4716 ;
                     14A8  4717 ; FUNCTIONAL DESCRIPTION:
                     14A8  4718 ;
                     14A8  4719 ;       This routine compares the display line number stored in the system block
                     14A8  4720 ;       at SHC$SB$B_DIS_POS  (SB$W_SIZE+1)  with the display line number stored
                     14A8  4721 ;       during the last update.  If different it indicates that the data for
                     14A8  4722 ;       this system has moved to a different part of the display.  If the data
                     14A8  4723 ;       itself has has not changed, then this could otherwise go undetected.
                     14A8  4724 ;
                     14A8  4725 ; CALLING SEQUENCE:
                     14A8  4726 ;
                     14A8  4727 ;       JSB     CHECK_POS
                     14A8  4728 ;
                     14A8  4729 ; CALLED BY:
                     14A8  4730 ;
                     14A8  4731 ;       CLUSTER_REPORT
                     14A8  4732 ;
                     14A8  4733 ; INPUTS:
                     14A8  4734 ;
                     14A8  4735 ;       R11 = Base address of system block containing line number to check.
                     14A8  4736 ;       OLD_VAR_BUF points to buffer containing system blocks from the previous
                     14A8  4737 ;       pass.
                     14A8  4738 ;
                     14A8  4739 ; OUTPUTS:
                     14A8  4740 ;
                     14A8  4741 ;       If the line numbers are found to be different, then a bit corresponding
                     14A8  4742 ;       to the line number is set in LINE_DIF.
                     14A8  4743 ;
                     14A8  4744 ; RETURN STATUS:
                     14A8  4745 ;
                     14A8  4746 ;       R0 = SS$_NORMAL
                     14A8  4747 ;
                     14A8  4748 ;--
                     14A8  4749
                     14A8  4750 CHECK_POS:
                 0C  BB  14A8  4751          PUSHR   #^M<R2,R3>
000001F3'8F      5B  D1  14AA  4752          CMPL    R11,#DUMMY_SB           ; Does R11 point to the fake system
                     14B1  4753                                              ;  block?
                 14  12  14B1  4754          BNEQ    10$                     ; Branch if not.
53   000004A3'EF     DE  14B3  4755          MOVAL   OLD_FIX_BUF,R3          ; Point R3 to previous fixed length
                     14BA  4756                                              ;  buffers.
52   0000004B'EF     DE  14BA  4757          MOVAL   FIX_BUF,R2              ; Get the address of the current fixed
                     14C1  4758                                              ;  length buffers.
     52  5B  52     C3  14C1  4759          SUBL3   R2,R11,R2               ; Get the offset into the current
                     14C5  4760                                              ;  fixed length buffer where this
                     14C5  4761                                              ;  system block is located.
                 18  11  14C5  4762          BRB     20$                     ; Branch to common.
53   00000419'EF     D0  14C7  4763 10$:     MOVL    OLD_VAR_BUF,R3          ; Point R3 to the previous variable
                     14CE  4764                                              ;  length buffers.
52   5B  000003ED'EF     C3  14CE  4765          SUBL3   VAR_BUF,R11,R2          ; Get the offset into the current
                     14D6  4766                                              ;  variable length buffer where this
                     14D6  4767                                              ;  system block is located.
00000421'EF      52  D1  14D6  4768          CMPL    R2,OLD_VAR_BUF_LEN      ; Is the offset larger than length of
                     14DD  4769                                              ;  the previous buffer?
                 0A  18  14DD  4770          BGEQ    30$                     ; Branch if so.
             53  52  C0  14DF  4771 20$:     ADDL    R2,R3                   ; Locate the system block in the
```

```
                                14E2  4772                                      ;  old buffer.
                          09 AB 91 14E2  4773          CMPB    SHC$SB$B_DIS_POS(R11),- ;  Is the line number different?
                          09 A3    14E5  4774                  SHC$SB$B_DIS_POS(R3)    ;
                             0C 13 14E7  4775          BEQL    40$                     ;  Branch if not.
  00 00000434'EF  00002248'EF E2 14E9  4776 30$:      BBSS    LINE_NUM,LINE_DIF,40$   ;  Indicate line must be redrawn.
                             0C BA 14F5  4777 40$:      POPR    #^M<R2,R3>
                                05 14F7  4778          RSB
                                   14F8  4779
                                   14F8  4780
```

```
                     14F8  4782 ;++
                     14F8  4783 ;
                     14F8  4784 ; FUNCTIONAL DESCRIPTION:
                     14F8  4785 ;
                     14F8  4786 ;       This routine takes the system ID found in the system block and calls
                     14F8  4787 ;       FIND_SYS_ID to check if the system ID is in the SYS_ID_LIST of system
                     14F8  4788 ;       that are not to be display.  It returns success if the system ID was
                     14F8  4789 ;       found.
                     14F8  4790 ;
                     14F8  4791 ; CALLING SEQUENCE:
                     14F8  4792 ;
                     14F8  4793 ;       JSB     CHECK_ID
                     14F8  4794 ;
                     14F8  4795 ; CALLED BY:
                     14F8  4796 ;
                     14F8  4797 ;       CLUSTER_REPORT
                     14F8  4798 ;
                     14F8  4799 ; INPUTS:
                     14F8  4800 ;
                     14F8  4801 ;       R11 = Base address of system block containing ID to check.
                     14F8  4802 ;
                     14F8  4803 ;
                     14F8  4804 ; RETURN STATUS:
                     14F8  4805 ;
                     14F8  4806 ;       R0 = SS$_NORMAL if system ID is in the list.
                     14F8  4807 ;
                     14F8  4808 ;--
                     14F8  4809
                     14F8  4810 CHECK_ID:
          10   BB    14F8  4811          PUSHR   #^M<R4>                 ; R4 destroyed by FIND_SYS_ID.
        18 AB  D0    14FA  4812          MOVL    SB$B_SYSTEMID(R11),-    ; Get the system ID from the system
000022DC'EF         14FD  4813                  ID_RESULT               ;  block and store in ID_RESULT.
        1C AB  B0    1502  4814          MOVW    SB$B_SYSTEMID+4(R11),-  ;
000022E0'EF         1505  4815                  ID_RESULT+4             ;
00000E16'EF   16    150A  4816          JSB     FIND_SYS_ID             ; Look for it in the list.
          10   BA    1510  4817          POPR    #^M<R4>                 ; Restore R4.
               05    1512  4818          RSB                             ; Return with status from FIND_SYS_ID.
                     1513  4819
                     1513  4820
```

```
                     1513  4822          .SBTTL  Local report generation routine.
                     1513  4823 ;++
                     1513  4824 ;
                     1513  4825 ; FUNCTIONAL DESCRIPTION:
                     1513  4826 ;
                     1513  4827 ;       This routine formats the display for the local port report.  It simply
                     1513  4828 ;       enables both classes in the local report gets the address of the
                     1513  4829 ;       information for the port PDT_BUF and MISC_BUF and calls the GET_DATA
                     1513  4830 ;       routine.  It then determines if a second report is present and if so
                     1513  4831 ;       repeats the process for the second report.
                     1513  4832 ;
                     1513  4833 ;       Note that V_SCROLL_LINE is cleared for this report since it is never
                     1513  4834 ;       necessary to scroll this report vertically.
                     1513  4835 ;
                     1513  4836 ; CALLING SEQUENCE:
                     1513  4837 ;
                     1513  4838 ;       JSB     LOCAL_REPORT
                     1513  4839 ;
                     1513  4840 ; CALLED BY:
                     1513  4841 ;
                     1513  4842 ;       MAIN_LOOP
                     1513  4843 ;
                     1513  4844 ; INPUTS:
                     1513  4845 ;
                     1513  4846 ;       CLASS_SELECT    Contains a mask of the classes selected for display
                     1513  4847 ;                       by the user.
                     1513  4848 ;
                     1513  4849 ; IMPLICIT INPUTS:
                     1513  4850 ;
                     1513  4851 ;       Snapshot has been performed and all the port discriptors and
                     1513  4852 ;       miscellaneous port information has been stored in SNAP_BUF.
                     1513  4853 ;
                     1513  4854 ; IMPLICIT OUTPUTS:
                     1513  4855 ;
                     1513  4856 ;       Display data stored in SCREEN_BUF.
                     1513  4857 ;
                     1513  4858 ; RETURN STATUS:
                     1513  4859 ;
                     1513  4860 ;       Status from call to GET_DATA routine passed in R0.
                     1513  4861 ;
                     1513  4862 ; REGISTER USAGE:
                     1513  4863 ;
                     1513  4864 ;       R9  -   Pointer into MISC_BUF.
                     1513  4865 ;       R10 -   Pointer into PDT_BUF.
                     1513  4866 ;
                     1513  4867 ;--
                     1513  4868
                     1513  4869 LOCAL_REPORT:
         0600 8F BB  1513  4870          PUSHR   #^M<R9,R10>
5A   000002A7'EF DE  1517  4871          MOVAL   PDT_BUF,R10             ; Get address of first PDT.
59   00000447'EF DE  151E  4872          MOVAL   MISC_BUF,R9             ; Get address of miscellaneous buffer.
     00000180 8F DD  1525  4873          MOVL    #CLS$M_LPRT!CLS$M_ERR,- ; Enable display of local ports and
     000021F8'EF     152B  4874                  DATA_CNTRL              ;  errors.
            69 D5    1530  4875          TSTL    (R9)                    ; Any data on first port?
            05 13    1532  4876          BEQL    10$                     ; Branch if not.
            14 10    1534  4877          BSB     100$                    ; Get data for this port.
          0C 50 E9   1536  4878          BLBC    R0,99$                  ; Branch on error.
```

```
                59   2E   C0  1539  4879 10$:     ADDL    #MISC_BUF_LEN,R9          ; Get address of miscellaneous data for
                                1539  4880                                          ;  the second port.
     5A 000000D0 8F   C0  153C  4881             ADDL    #SHC$L_PDT_LENGTH,R10     ; Get address of PDT for second port.
                     05   10  1543  4882             BSB     100$                      ; Get data for this port.
                0600 8F   BA  1545  4883 99$:      POPR    #^M<R9,R10>
                     05   1549  4884             RSB
                                154A  4885
                     69   D5  154A  4886 100$:     TSTL    (R9)                      ; Any data on port?
                     0E   13  154C  4887             BEQL    199$                      ; If not return.
                     59   DD  154E  4888             PUSHL   R9                        ; Save address of miscellaneous buffer.
                     5A   DD  1550  4889             PUSHL   R10                       ; Save address of PDT
                     5E   14  C2  1552  4890             SUBL    #5*4,SP                   ; Adjust SP for no CDT, PB, CSB, SB, or
                                1555  4891                                          ;  CLUB.
00002729'EF          07   FB  1555  4892             CALLS   #7,GET_DATA               ; Get info on port.
                     05   155C  4893 199$:     RSB
                                155D  4894
                                155D  4895
```

```
                        155D  4897          .SBTTL  Error message output.
                        155D  4898 ;++
                        155D  4899 ;
                        155D  4900 ; FUNCTIONAL DESCRIPTION:
                        155D  4901 ;
                        155D  4902 ;       This routine does a call to $GETMSG using the status stored in
                        155D  4903 ;       CMD_STATUS. (normally the result from the last command parse).
                        155D  4904 ;       It then determines if any FAO arguments are expected with the
                        155D  4905 ;       message and sets up the message vector accordingly.  Finally it
                        155D  4906 ;       does a call to LIB$SIGNAL to output the message to the terminal.
                        155D  4907 ;
                        155D  4908 ; CALLING SEQUENCE:
                        155D  4909 ;
                        155D  4910 ;       JSB     SIGNAL
                        155D  4911 ;
                        155D  4912 ; CALLED BY:
                        155D  4913 ;
                        155D  4914 ;       READ_W_PROMPT
                        155D  4915 ;       SCROLL_TWO
                        155D  4916 ;
                        155D  4917 ; INPUTS:
                        155D  4918 ;
                        155D  4919 ;       Message identification number stored at CMD_STATUS.
                        155D  4920 ;
                        155D  4921 ; IMPLICIT IMPUTS:
                        155D  4922 ;
                        155D  4923 ;       Descriptor for FAO argument stored at TPARSE_BLK+TPA$L_STRINGCNT.
                        155D  4924 ;
                        155D  4925 ; OUTPUTS:
                        155D  4926 ;
                        155D  4927 ;       Message output to the terminal.
                        155D  4928 ;
                        155D  4929 ; RETURN STATUS:
                        155D  4930 ;
                        155D  4931 ;       SS$_NORMAL or failed status returned in R0.
                        155D  4932 ;
                        155D  4933 ;--
                        155D  4934
                        155D  4935 SIGNAL:
                000004B9'EF   D1  155D  4936          CMPL    CMD_STATUS,-            ; Syntax error detected by TPARSE?
                00000000'8F       1563  4937                  #LIB$_SYNTAXERR         ;
                          0D  12  1568  4938          BNEQ    10$                     ; Branch if not.
000004B9'EF     00000000'8F   D0  156A  4939          MOVL    #SHOW$_UNRCMD,CMD_STATUS; "Unrecognized command"
                          32  11  1575  4940          BRB     30$                     ; Branch to common code.
                              1577  4941
00000000'8F     000004B9'EF   D1  1577  4942 10$:     CMPL    CMD_STATUS,#LIB$_UNRKEY ; Unrecognized keyword?
                          0D  12  1582  4943          BNEQ    20$                     ; Branch if not.
000004B9'EF     00000000'8F   D0  1584  4944          MOVL    #SHOW$_UNRKEY,CMD_STATUS; "Unrecognized keyword"
                          18  11  158F  4945          BRB     30$                     ; Branch to common code.
                              1591  4946
00000000'8F     000004B9'EF   D1  1591  4947 20$:     CMPL    CMD_STATUS,#LIB$_AMBKEY ; Ambiguous keyword?
                          0B  12  159C  4948          BNEQ    30$                     ; Branch if not.
000004B9'EF     00000000'8F   D0  159E  4949          MOVL    #SHOW$_AMBIG,CMD_STATUS ; "Ambiguous keyword"
                              15A9  4950
                              15A9  4951 30$:     $GETMSG_S -                         ; Get the error message into the
                              15A9  4952                  MSGID = CMD_STATUS,-    ;  buffer.
                              15A9  4953                  MSGLEN = ERR_MSG_DSCR,- ;
```

```
                         15A9  4954                    BUFADR = GET_MSG_DSCR,- ;
                         15A9  4955                    OUTADR = MSG_OUT_ADRS   ;
                   4C 50 E9   15CA  4956          BLBC    R0,99$                  ; Branch on error.
                         15CD  4957
        000004AE'EF    95   15CD  4958          TSTB    MSG_OUT_ADRS+1          ; Is there an FAO argument associated
                         15D3  4959                                                ;  with the message.
                      2E 13   15D3  4960          BEQL    40$                     ; Branch if not.
  0000049D'EF         03 B0   15D5  4961          MOVW    #3,ERR_MSGVEC           ; Otherwise set the number of longwords
                         15DC  4962                                                ;  in the message vector to include the
                         15DC  4963                                                ;  FAO argument.
        000004AE'EF    90   15DC  4964          MOVB    MSG_OUT_ADRS+1,-        ; Save the number of FAO arguments in
        000004A5'EF         15E2  4965                  ERR_MSGVEC+8            ;  the message vector.
        000004A9'EF    DD   15E7  4966          PUSHL   ERR_MSGVEC+12           ;
  7E    000004A5'EF    3C   15ED  4967          MOVZWL  ERR_MSGVEC+8,-(SP)      ; Output the error message with FAO
        000004B9'EF    DD   15F4  4968          PUSHL   CMD_STATUS              ;  directives.
  00000000'GF          03 FB   15FA  4969          CALLS   #3,G^LIB$SIGNAL         ;
                      16 11   1601  4970          BRB     99$                     ; Branch to common code.
                         1603  4971
  0000049D'EF         01 90   1603  4972 40$:     MOVB    #1,ERR_MSGVEC           ; Set the number of longwords in the
                         160A  4973                                                ;  message vector to indicate that
                         160A  4974                                                ;  there are no FAO arguments are
                         160A  4975                                                ;  associated with the message.
        000004B9'EF    DD   160A  4976          PUSHL   CMD_STATUS              ; Output the error message.
  00000000'GF          01 FB   1610  4977          CALLS   #1,G^LIB$SIGNAL         ;
                      00 11   1617  4978          BRB     99$                     ; Branch to common code.
                         1619  4979
                      05   1619  4980 99$:     RSB
                         161A  4981
                         161A  4982
```

```
                     161A  4984          .SBTTL  Broadcast message handling.
                     161A  4985 ;++
                     161A  4986 ;
                     161A  4987 ; FUNCTIONAL DESCRIPTION:
                     161A  4988 ;
                     161A  4989 ;       This routine is called to output a broadcast message.  It determines
                     161A  4990 ;       if any information is being display on the bottom line of the report
                     161A  4991 ;       (command input, error messages, or a previous broadcast message).
                     161A  4992 ;       If no information is being displayed, then a line of a broadcast
                     161A  4993 ;       message is removed from the MSG_OUT_QUE (if available) and displayed
                     161A  4994 ;       on the bottom line.  Once output, the buffer used for the message is
                     161A  4995 ;       placed in the MSG_FRE_QUE to save calls to GET_VM by the STORE_MSG
                     161A  4996 ;       routine.
                     161A  4997 ;
                     161A  4998 ; CALLING SEQUENCE:
                     161A  4999 ;
                     161A  5000 ;       JSB     OUT_MSG
                     161A  5001 ;
                     161A  5002 ; CALLED BY:
                     161A  5003 ;
                     161A  5004 ;       CHECK_MORE
                     161A  5005 ;
                     161A  5006 ; IMPLICIT INPUTS:
                     161A  5007 ;
                     161A  5008 ;       BOT_CMD_LINE    Indicates the type of informatior currently being
                     161A  5009 ;                       displayed in the bottom line of the command scroll
                     161A  5010 ;                       region.
                     161A  5011 ;
                     161A  5012 ;       MSG_OUT_QUE     Contains a list of buffers containing message lines
                     161A  5013 ;                       to be output.
                     161A  5014 ;
                     161A  5015 ; IMPLICIT OUTPUTS:
                     161A  5016 ;
                     161A  5017 ;       Broadcast message output to the screen buffer.
                     161A  5018 ;       Message buffer added to MSG_FRE_QUE
                     161A  5019 ;       Message buffer removed from MSG_OUT_QUE
                     161A  5020 ;
                     161A  5021 ;       BOT_CMD_LINE    Set to indicate that there is a broadcast message on
                     161A  5022 ;                       the bottom line of the command scroll region if a
                     161A  5023 ;                       message was output.
                     161A  5024 ;
                     161A  5025 ; RETURN STATUS:
                     161A  5026 ;
                     161A  5027 ;       SS$_NORMAL or failed status returned in R0.
                     161A  5028 ;
                     161A  5029 ;--
                     161A  5030
                  3C BB  161A  5031 OUT_MSG:PUSHR   #^M<R2,R3,R4,R5>
      00000471'EF 95  161C  5032          TSTB    BOT_CMD_LINE            ; Any information on the bottom command
                     1622  5033                                          ;  line?
                     1622  5034          JNEQ    99$                     ; Branch if so.
00000000'EF 00000010'FF 0F  1627  5035          REMQUE  @MSG_OUT_QUE,BUFADR     ; Remove a line from the message output
                     1632  5036                                          ;  queue.
                  03 1C  1632  5037          BVC     10$                     ; Skip next if available.
                007F 31  1634  5038          BRW     99$                     ; Branch if none available.
   54 00000000'EF 08 C1  1637  5039 10$:     ADDL3   #8,BUFADR,R4            ; Get the address of the message.
                     163F  5040                                          ; Build message descriptor
```

```
                00001FAD'EF    84    3C  163F  5041          MOVZWL  (R4)+,MBX_MSG_DSCR      ;  - length and type
                00001FB1'EF    54    D0  1646  5042          MOVL    R4,MBX_MSG_DSCR+4      ;  - address.
     00002248'EF    00002260'EF    D0  164D  5043          MOVL    LAST_LINE,LINE_NUM     ; Output message on the bottom line.
                    0000224C'EF    3F  1658  5044          PUSHAW  COL_NUM                ; Output the broadcast message.
                    00002248'EF    3F  165E  5045          PUSHAW  LINE_NUM               ;
                    00001FAD'EF    3F  1664  5046          PUSHA   MBX_MSG_DSCR           ;
                00000000'GF    03    FB  166A  5047          CALLS   #3,G^LIB$PUT_SCREEN    ;
                          42 50    E9  1671  5048          BLBC    R0,99$                 ; Branch on error.
                00000000'GF    00    FB  1674  5049          CALLS   #0,G^LIB$ERASE_LINE    ; Erase the rest of the line.
                          38 50    E9  167B  5050          BLBC    R0,99$                 ; Branch on error.
             1B 00002247'EF    01    E0  167E  5051          BBS     #DVT$V_DCRT,OUT_DEV,20$ ; Branch if DECcrt terminal.
                    00001FAD'EF    28  1686  5052          MOVC3   MBX_MSG_DSCR,-         ; Save a copy of the last message
                    00001FB1'EF        168C  5053                  MBX_MSG_DSCR+4,-       ;  output so that in can be scrolled.
                    00001DFA'EF        1691  5054                  LAST_BRDCST_BUF        ;
                    00001FAD'EF    D0  1696  5055          MOVL    MBX_MSG_DSCR,-         ; Save the length of the message.
                    00001DF2'EF        169C  5056                  LAST_BRDCST_DSCR       ;
                              01    88  16A1  5057 20$:     BISB    #CLI$M_BRDCST,-        ; Indicate that there is a broadcast
                    00000471'EF        16A3  5058                  BOT_CMD_LINE           ;  message on the bottom command line.
                          50    01    D0  16A8  5059          MOVL    #SS$_NORMAL,R0         ;
     00000008'EF    00000000'FF    0E  16AB  5060          INSQUE  @BUFADR,MSG_FRE_QUE    ; Place message buffer in message free
                                       16B6  5061                                         ;  queue.
                              3C    BA  16B6  5062 99$:     POPR    #^M<R2,R3,R4,R5>       ;
                                    05  16B8  5063          RSB                            ;
                                        16B9  5064                                         ;
                                        16B9  5065                                         ;
```

```
                        16B9  5067          .SBTTL  Scrolling routines.
                        16B9  5068          .SBTTL          -       SET_SCROLL routine.
                        16B9  5069
                        16B9  5070          .ENABL  LSB
                        16B9  5071
                        16B9  5072 ;++
                        16B9  5073 ;
                        16B9  5074 ; FUNCTIONAL DESCRIPTION:
                        16B9  5075 ;
                        16B9  5076 ;       This routine translates the characters following an escape in the
                        16B9  5077 ;       command buffer into a scroll command.  If the escape sequence indicates
                        16B9  5078 ;       that one of the arrow keys has been pressed then the value stored at
                        16B9  5079 ;       location V_SCROLL (for vertical scrolling) or either CLSTR_H_SCROLL
                        16B9  5080 ;       or LOCAL_H_SCROLL (for horizontal scrolling depending on which report
                        16B9  5081 ;       is being displayed) is incremented or decremented appropriately.
                        16B9  5082 ;
                        16B9  5083 ;       This routine also sets flags at location SCROLL_FLAGS if it determines
                        16B9  5084 ;       that data has been scrolled off the left or top of the screen or clears
                        16B9  5085 ;       the flags if the data has been restored.
                        16B9  5086 ;
                        16B9  5087 ;       If the terminal type is DECcrt, and the display is scrolled up or
                        16B9  5088 ;       down, then bits are set in SCROLL_DIF corresponding to the lines
                        16B9  5089 ;       which much be redrawn to complete the display.
                        16B9  5090 ;
                        16B9  5091 ;       NOTE:   This routine does not set or clear the flags that indicate that
                        16B9  5092 ;               the data has been truncated at the right side or bottom of the
                        16B9  5093 ;               screen
                        16B9  5094 ;
                        16B9  5095 ; CALLING SEQUENCE:
                        16B9  5096 ;
                        16B9  5097 ;       JSB     SET_SCROLL
                        16B9  5098 ;
                        16B9  5099 ; CALLED BY:
                        16B9  5100 ;
                        16B9  5101 ;       READ_WO_PROMPT
                        16B9  5102 ;
                        16B9  5103 ; IMPLICIT INPUTS:
                        16B9  5104 ;
                        16B9  5105 ;       Escape sequence stored in CMD_BUF.
                        16B9  5106 ;
                        16B9  5107 ; OUTPUTS:
                        16B9  5108 ;
                        16B9  5109 ;       V_SCROLL, CLSTR_H_SCROLL, or LOCAL_H_SCROLL updated.
                        16B9  5110 ;       SCROLL_FLAGS updated.
                        16B9  5111 ;       SCROLL_DIF updated.
                        16B9  5112 ;
                        16B9  5113 ;--
                        16B9  5114
                        16B9  5115 SET_SCROLL:
9B 8F   00002242'EF  91 16B9  5116          CMPB    IOSB+4,#CSI             ; Did the user type an eight-bit escape
                        16C1  5117                                          ;  sequence?
                    11  12 16C1  5118          BNEQ    10$                     ; Branch if not.
000020DB'EF   5B 8F  90 16C3  5119          MOVB    #^A/[/,CMD_BUF          ; Substitute an open bracket.
   51   000020DB'EF  DE 16CB  5120          MOVAL   CMD_BUF,R1              ; Point to open bracket.
                    07  11 16D2  5121          BRB     20$                     ; Jump to common code.
   51   000020DC'EF  DE 16D4  5122 10$:     MOVAL   CMD_BUF+1,R1            ; Point to open bracket.
   00002247'EF   01  E1 16DB  5123 20$:     BBC     #DVT$V_DCRT,OUT_DEV,-   ; Branch if not DECcrt terminal.
```

```
                     36        16E2  5124                    40$:
 445B 8F    61    B1  16E3  5125          CMPW    (R1),#DCRT_RT_ARROW     ; Right arrow?
                      16E8  5126          JEQL    DEC_H_SCROLL            ; Scroll right.
 435B 8F    61    B1  16ED  5127          CMPW    (R1),#DCRT_LFT_ARROW    ; Left arrow?
                      16F2  5128          JEQL    INC_H_SCROLL            ; Scroll left.
              00    E0  16F7  5129          BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
03 00000000'EF                16F9  5130                  CURRENT_REPORT,30$  ;  cluster report.
            01C6    31  16FF  5131          BRW     999$                    ; Otherwise, don't allow up or down
                        1702  5132                                          ;  scroll.
 415B 8F    61    B1  1702  5133  30$:    CMPW    (R1),#DCRT_UP_ARROW     ; Up arrow?
                      1707  5134          JEQL    DEC_V_SCROLL            ; Scroll up.
 425B 8F    61    B1  170C  5135          CMPW    (R1),#DCRT_DWN_ARROW    ; Down arrow?
                      1711  5136          JEQL    INC_V_SCROLL            ; Scroll down.
            01AF    31  1716  5137          BRW     999$                    ; Must be invalid escape sequence.
                        1719  5138
   44 8F    61    91  1719  5139  40$:    CMPB    (R1),#VT52_RT_ARROW     ; Right arrow?
                      171D  5140          JEQL    DEC_H_SCROLL            ; Scroll right.
   43 8F    61    91  1722  5141          CMPB    (R1),#VT52_LFT_ARROW    ; Left arrow?
                      1726  5142          JEQL    INC_H_SCROLL            ; Scroll left.
              00    E0  172B  5143          BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
03 00000000'EF                172D  5144                  CURRENT_REPORT,50$  ;  cluster report.
            0192    31  1733  5145          BRW     999$                    ; Otherwise, don't allow up or down
                        1736  5146                                          ;  scroll.
   41 8F    61    91  1736  5147  50$:    CMPB    (R1),#VT52_UP_ARROW     ; Up arrow?
              0C    13  173A  5148          BEQL    DEC_V_SCROLL            ; Scroll up.
   42 8F    61    91  173C  5149          CMPB    (R1),#VT52_DWN_ARROW    ; Down arrow?
                      1740  5150          JEQL    INC_V_SCROLL            ; Scroll down.
            0180    31  1745  5151          BRW     999$                    ; Must be invalid escape sequence.
                        1748  5152
                        1748  5153  DEC_V_SCROLL:
          0000045C'EF   D5  1748  5154          TSTL    V_SCROLL                ; Already scrolled all the way down?
                      174E  5155          JEQL    999$                    ; If so, don't scroll.
          00000430'EF   D6  1753  5156          INCL    SCROLL_LINES            ; Increase number of times we are
                        1759  5157                                          ;  allowed to scroll up.
          0000045C'EF   D7  1759  5158          DECL    V_SCROLL                ; Scroll display down.
                  07    12  175F  5159          BNEQ    60$                     ; Branch if we're not all the way down.
                  04    8A  1761  5160          BICB    #SCR$M_MORE_UP,-        ; Otherwise, clear the up flag.
          0000047A'EF       1763  5161                  SCROLL_FLAGS            ;
       03 00002247'EF   01    E0  1768  5162  60$:    BBS     #DVT$V_DCRT,OUT_DEV,70$ ; Branch if terminal is a DECcrt.
                0152    31  176A  5163          BRW     140$                    ; Otherwise branch to set refresh flag.
                0266    30  176D  5164  70$:    BSBW    SET_REGION_DATA         ; Set scrolling region to include only
                        1770  5165                                          ;  the data area of the screen.
   00002248'EF  00002254'EF   D0  1770  5166          MOVL    FIRST_DATA_LINE,LINE_NUM; Set the cursor to the first line of
                0000224C'EF   3F  1775  5167          PUSHAW  COL_NUM                 ;  data.
                00002248'EF   3F  177B  5168          PUSHAW  LINE_NUM                ;
            00000000'GF   02    FB  1781  5169          CALLS   #2,G^LIB$SET_CURSOR     ;
                        1788  5170          JLBC    R0,999$                 ; Branch on error.
            00000000'GF   00    FB  178E  5171          CALLS   #0,G^LIB$DOWN_SCROLL    ; Scroll down one line.
                        1795  5172          JLBC    R0,999$                 ; Branch on error.
                0224    30  179B  5173          BSBW    SET_REGION_FULL         ; Reset the scrolling region to use the
                        179E  5174                                          ;  full screen.
00000444'EF 00000444'EF   01    9C  179E  5175          ROTL    #1,SCROLL_DIF,SCROLL_DIF; Force the top line of data to be
                00002254'EF   E2  17AA  5176          BBSS    FIRST_DATA_LINE,-       ;  redrawn.
                  00 00000444'EF  17B0  5177                  SCROLL_DIF,80$          ;
                0103    31  17B6  5178  80$:    BRW     999$                    ; Finish up.
                        17B9  5179
                        17B9  5180  INC_V_SCROLL:
```

```
                0000007E 8F    D3  17C5  5181          BITL    #CLS$M_ANY_SCS,-        ; Any SCS fields being displayed?
                00000001'EF        17CB  5182                  CLASS_SELECT            ;
                                   17D0  5183          JEQL    999$                    ; Branch if not.
                                   17D5  5184 ;
                                   17D5  5185 ; Since we don't really know how many lines might be displayed in a given
                                   17D5  5186 ; report, we don't really know how many times we can up-scroll.  Because of
                                   17D5  5187 ; this, we limit the number of times the user can hit the up arrow before
                                   17D5  5188 ; we get a chance to redraw the screen and see how many more lines we have.
                                   17D5  5189 ;
                00000430'EF    D5  17D5  5190          TSTL    SCROLL_LINES            ; Any more lines to be scrolled?
                                   17DB  5191          JEQL    999$                    ; Branch if not.
                00000430'EF    D7  17E0  5192          DECL    SCROLL_LINES            ; Now there's one less.
                0000045C'EF    D6  17E6  5193          INCL    V_SCROLL                ; Scroll display up.
          58 00002247'EF    01   E1  17EC  5194          BBC     #DVT$V_DCRT,OUT_DEV,90$ ; Branch if terminal isn't a DECcrt.
                        01E5    30  17F4  5195          BSBW    SET_REGION_DATA         ; Set scrolling region to include only
                                   17F7  5196                                          ;  the data area of the screen.
    00002248'EF 00002258'EF    D0  17F7  5197          MOVL    LAST_DATA_LINE,LINE_NUM ; Set the cursor to the last line of
                0000224C'EF    3F  1802  5198          PUSHAW  COL_NUM                 ;  data.
                00002248'EF    3F  1808  5199          PUSHAW  LINE_NUM                ;
             00000000'GF    02  FB  180E  5200          CALLS   #2,G^LIB$SET_CURSOR     ;
                                   1815  5201          JLBC    R0,999$                 ; Branch on error.
             00000000'GF    00  FB  181B  5202          CALLS   #0,G^LIB$UP_SCROLL      ; Scroll up one line.
                                   1822  5203          JLBC    R0,999$                 ; Branch on error.
                        01A3    30  1828  5204          BSBW    SET_REGION_FULL         ; Reset the scrolling region to use the
                                   182B  5205                                          ;  full screen.
                          03    E1  182B  5206          BBC     #SCR$V_MORE_DWN,-       ; Branch if the bottom line doesn't
             19 0000047A'EF         182D  5207                  SCROLL_FLAGS,90$        ;  need to be drawn.
          00000444'EF    FF 8F    9C  1833  5208          ROTL    #-1,SCROLL_DIF,-        ; Force the last data line to be
                00000444'EF        183B  5209                  SCROLL_DIF              ;  drawn.
                00002258'EF    E2  1840  5210          BBSS    LAST_DATA_LINE,-        ;
             00 00000444'EF         1846  5211                  SCROLL_DIF,90$          ;
                          14    88  184C  5212 90$:     BISB    #SCR$M_MORE_UP!SCR$M_MORE,-
                0000047A'EF        184E  5213                  SCROLL_FLAGS            ; Indicate screen can be scrolled down.
          66 00002247'EF    01   E1  1853  5214          BBC     #DVT$V_DCRT,OUT_DEV,140$; Branch if terminal isn't a DECcrt.
                          6B    11  185B  5215          BRB     999$                    ; Otherwise, don't set the refresh flag.
                                   185D  5216
                                   185D  5217 INC_H_SCROLL:
                00000425'EF    95  185D  5218          TSTB    NUM_FIELDS              ; Any fields to be displayed?
                          63    13  1863  5219          BEQL    999$                    ; Branch if not.
                          00    E1  1865  5220          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
             08 00000000'EF         1867  5221                  CURRENT_REPORT,100$     ;  cluster report.
                00000464'EF    D6  186D  5222          INCL    CLSTR_H_SCROLL          ; Otherwise scroll the cluster report
                                   1873  5223                                          ;  left.
                          06    11  1873  5224          BRB     110$                    ; Branch to common code.
                00000468'EF    D6  1875  5225 100$:     INCL    LOCAL_H_SCROLL          ; Scroll the local report left.
                00000425'EF    97  187B  5226 110$:     DECB    NUM_FIELDS              ; Decrease number of times we can scroll
                                   1881  5227                                          ;  left.
                          12    88  1881  5228          BISB    #SCR$M_MORE_LFT!SCR$M_MORE,-
                0000047A'EF        1883  5229                  SCROLL_FLAGS            ; Indicate screen can be scrolled
                                   1888  5230                                          ;  right.
                          37    11  1888  5231          BRB     140$                    ; Done.
                                   188A  5232
                                   188A  5233 DEC_H_SCROLL:
                          00    E1  188A  5234          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
             12 00000000'EF         188C  5235                  CURRENT_REPORT,120$     ;  cluster report.
                00000464'EF    D5  1892  5236          TSTL    CLSTR_H_SCROLL          ; Is the cluster report scrolled all
                                   1898  5237                                          ;  the way to the right?
```

```
                     2E   13  1898  5238          BEQL    999$                        ; Don't scroll anymore if so.
          00000464'EF     D7  189A  5239          DECL    CLSTR_H_SCROLL              ; Otherwise scroll the cluster report
                          18A0  5240                                                  ;  right.
                     19   12  18A0  5241          BNEQ    135$                        ; Branch if the cluster report isn't
                          18A2  5242                                                  ;  scrolled all the way to the right.
                     10   11  18A2  5243          BRB     130$                        ; Branch to common code.
                          18A4  5244
          00000468'EF     D5  18A4  5245 120$:     TSTL    LOCAL_H_SCROLL              ; Is the local report scrolled all the
                          18AA  5246                                                  ;  way to the right?
                     1C   13  18AA  5247          BEQL    999$                        ; Don't scroll anymore if so.
          00000468'EF     D7  18AC  5248          DECL    LOCAL_H_SCROLL              ; Otherwise, scroll the local report
                          18B2  5249                                                  ;  right.
                     07   12  18B2  5250          BNEQ    135$                        ; Branch if the local report isn't
                          18B4  5251                                                  ;  scrolled all the way to the right.
                     02   8A  18B4  5252 130$:     BICB    #SCR$M_MORE_LFT,-           ; Clear left flag.
          0000047A'EF         18B6  5253                  SCROLL_FLAGS                ;
          00000425'EF     96  18BB  5254 135$:     INCB    NUM_FIELDS                  ; Increase number of times we can scroll
                          18C1  5255                                                  ;  left.
     00000476'EF     08   C8  18C1  5256 140$:     BISL    #MF$M_SKIP,MISC_FLAGS       ; Set flag to indicate that the screen
                          18C8  5257                                                  ;  should be refreshed.
00000476'EF  00000100 8F   C8  18C8  5258 999$:     BISL    #MF$M_SCROLL,MISC_FLAGS     ; Indicate we just scrolled.
                          05  18D3  5259          RSB
                              18D4  5260
                              18D4  5261          .DSABL  LSB
                              18D4  5262
                              18D4  5263
```

```
                                18D4  5265          .SBTTL          -       CHECK_MORE routine.
                                18D4  5266 ;++
                                18D4  5267 ;
                                18D4  5268 ; FUNCTIONAL DESCRIPTION:
                                18D4  5269 ;
                                18D4  5270 ;       This routine is used to output the "MORE" scrolling signals at the
                                18D4  5271 ;       bottom of the display.  It first calls the routine which outputs
                                18D4  5272 ;       broadcast messages (giving the messages priority over the scrolling
                                18D4  5273 ;       signals).  It then checks to see if any scroll flags are set.  If
                                18D4  5274 ;       there is a scroll flag set it then checks to see if any information has
                                18D4  5275 ;       been output on either of the command lines (the bottom two lines of the
                                18D4  5276 ;       display - the top line has priority).  If there is no information
                                16D4  5277 ;       indicated on one of the lines it adds the "MORE" scrolling signals to
                                18D4  5278 ;       the screen buffer.
                                18D4  5279 ;
                                18D4  5280 ; CALLING SEQUENCE:
                                18D4  5281 ;
                                18D4  5282 ;       JSB CHECK_MORE
                                18D4  5283 ;
                                18D4  5284 ; IMPLICIT INPUTS:
                                18D4  5285 ;
                                18D4  5286 ;       TOP_CMD_LINE and BOT_CMD_LINE indicate the type of data currently being
                                18D4  5287 ;             displayed in the command scroll region.
                                18D4  5288 ;
                                18D4  5289 ; IMPLICIT OUTPUTS:
                                18D4  5290 ;
                                18D4  5291 ;       Scrolling signals output to the screen buffer if no other data is being
                                18D4  5292 ;       displayed in the command scroll region.
                                18D4  5293 ;
                                18D4  5294 ; RETURN STATUS:
                                18D4  5295 ;
                                18D4  5296 ;       SS$_NORMAL or failed status returned in R0.
                                18D4  5297 ;--
                                18D4  5298
                                18D4  5299 CHECK_MORE:
                      FD43   30 18D4  5300          BSBW    OUT_MSG                 ; First see if there are any broadcast
                                18D7  5301                                          ;  messages to output.
                   04   00   EC 18D7  5302          CMPV    #SCR$V_MORE_RT,#4,-     ; Has any data been scrolled off the
            00   0000047A'EF      18DA  5303                  SCROLL_FLAGS,#0         ;  screen or truncated?
                                18E0  5304          JEQL    99$                     ; Branch if not.
                00000471'EF   95 18E5  5305          TSTB    BOT_CMD_LINE            ; Is the bottom command line blank?
                                18EB  5306          JNEQ    99$                     ; Branch if not.
00002248'EF   00002260'EF   D0 18F0  5307          MOVL    LAST_LINE,LINE_NUM      ; Output on the bottom command line.
    00002222'EF   0050 8F   B1 18FB  5308          CMPW    #NARROW,SCREEN_WIDTH    ; Is the terminal width 80 columns?
                      14   13 1904  5309          BEQL    5$                      ; Branch if it is.
                0000224C'EF   3F 1906  5310          PUSHAW  COL_NUM                 ; Output "                More   ".
                00002248'EF   3F 190C  5311          PUSHAW  LINE_NUM                ;
                00000090'EF   3F 1912  5312          PUSHAW  MORE_TABS               ;
                      12   11 1918  5313          BRB     7$                      ;
                0000224C'EF   3F 191A  5314 5$:      PUSHAW  COL_NUM                 ; Output "More  ".
                00002248'EF   3F 1920  5315          PUSHAW  LINE_NUM                ;
                00000078'EF   3F 1926  5316          PUSHAW  MORE_DSCR               ;
          00000000'GF   03   FB 192C  5317 7$:      CALLS   #3,G^LIB$PUT_SCREEN     ;
                                1933  5318          JLBC    R0,99$                  ; Branch on error.
                      04   93 1939  5319          BITB    #SCR$M_MORE_UP,-        ; Has the screen been scrolled up?
                0000047A'EF      193B  5320                  SCROLL_FLAGS            ;
                      12   13 1940  5321          BEQL    10$                     ; Branch if not.
```

```
                000000A7'EF   3F  1942  5322          PUSHAW  MORE_UP                 ; Output "^   ".
00000000'GF     01            FB  1948  5323          CALLS   #1,G^LIB$PUT_SCREEN     ;
                7B            50  E9    194F  5324    BLBC    R0,99$                 ; Branch on error.
                10                11    1952  5325    BRB     20$                    ; Jump to common code.
                000000D3'EF   3F  1954  5326 10$:     PUSHAW  NO_MORE                ; Output "    ".
0000000D'GF     01            FB  195A  5327          CALLS   #1,G^LIB$PUT_SCREEN     ;
                69            50  E9    1961  5328    BLBC    R0,99$                 ; Branch on error.
                08            93  1964  5329 20$:     BITB    #SCR$M_MORE_DWN,-      ; Are there more lines of data than can
                0000047A'EF       1966  5330                  SCROLL_FLAGS           ;  be displayed?
                12            13  196B  5331          BEQL    30$                    ; Branch if not.
                000000B2'EF   3F  196D  5332          PUSHAW  MORE_DWN               ; Output "v   ".
00000000'GF     01            FB  1973  5333          CALLS   #1,G^LIB$PUT_SCREEN     ;
                50            50  E9    197A  5334    BLBC    R0,99$                 ; Branch on error.
                0D                11    197D  5335    BRB     50$                    ; Jump to common code.
                000000D3'EF   3F  197F  5336 30$:     PUSHAW  NO_MORE                ; Output "    ".
00000000'GF     01            FB  1985  5337 40$:     CALLS   #1,G^LIB$PUT_SCREEN     ;
                02            93  198C  5338 50$:     BITB    #SCR$M_MORE_LFT,-      ; Has the screen been scrolled right?
                0000047A'EF       198E  5339                  SCROLL_FLAGS           ;
                12            13  1993  5340          BEQL    60$                    ; Branch if not.
                000000BD'EF   3F  1995  5341          PUSHAW  MORE_LFT               ; Output "<   ".
00000000'GF     01            FB  199B  5342          CALLS   #1,G^LIB$PUT_SCREEN     ;
                28            50  E9    19A2  5343    BLBC    R0,99$                 ; Branch on error.
                10                11    19A5  5344    BRB     70$                    ; Jump to common code.
                000000D3'EF   3F  19A7  5345 60$:     PUSHAW  NO_MORE                ; Output "    ".
00000000'GF     01            FB  19AD  5346          CALLS   #1,G^LIB$PUT_SCREEN     ;
                16            50  E9    19B4  5347    BLBC    R0,99$                 ; Branch on error.
                01            93  19B7  5348 70$:     BITB    #SCR$M_MORE_RT,-       ; Has the data been truncated?
                0000047A'EF       19B9  5349                  SCROLL_FLAGS           ;
                0D            13  19BE  5350          BEQL    99$                    ; Branch if not.
                000000C8'EF   3F  19C0  5351          PUSHAW  MORE_RT                ; Output ">".
00000000'GF     01            FB  19C6  5352          CALLS   #1,G^LIB$PUT_SCREEN     ;
                05            19CD  5353 99$:     RSB
                              19CE  5354
                              19CE  5355
```

```
                                                        I 11
                     19CE  5357          .SBTTL          -       Set scrolling region routines.
                     19CE  5358          .ENABL  LSB
                     19CE  5359 ;++
                     19CE  5360 ;
                     19CE  5361 ; FUNCTIONAL DESCRIPTION:
                     19CE  5362 ;
                     19CE  5363 ;       The followin four routines set the scrolling regions of the terminal.
                     19CE  5364 ;       The four regions are defined as follows:
                     19CE  5365 ;
                     19CE  5366 ;       FULL            The full screen
                     19CE  5367 ;       DATA            The line imediately following the header up to the
                     19CE  5368 ;                       command region.
                     19CE  5369 ;       COMMAND         The last two lines of the screen.
                     19CE  5370 ;       HELP            The line immediately following the header through the
                     19CE  5371 ;                       last line of the screen.
                     19CE  5372 ;
                     19CE  5373 ; CALLING SEQUENCE:
                     19CE  5374 ;
                     19CE  5375 ;       JSB     SET_REGION_FULL
                     19CE  5376 ;       JSB     SET_REGION_COMMAND
                     19CE  5377 ;       JSB     SET_REGION_DATA
                     19CE  5378 ;       JSB     SET_REGION_HELP
                     19CE  5379 ;
                     19CE  5380 ; CALLED BY:
                     19CE  5381 ;
                     19CE  5382 ;       READ_W_PROMPT
                     19CE  5383 ;       DEC_V_SCROLL
                     19CE  5384 ;       INC_V_SCROLL
                     19CE  5385 ;       SCROLL_TWO
                     19CE  5386 ;       GET_HELP
                     19CE  5387 ;
                     19CE  5388 ; RETURN STATUS:
                     19CE  5389 ;
                     19CE  5390 ;       SS$_NORMAL or failed status returned in R0.
                     19CE  5391 ;--
                     19CE  5392
                     19CE  5393 SET_REGION_FULL:
00002250'EF    3F    19CE  5394          PUSHAW  FIRST_LINE              ; Reset the scrolling region to use
00002260'EF    3F    19D4  5395          PUSHAW  LAST_LINE               ;  the full screen.
          3A   11    19DA  5396          BRB     10$                     ; Branch to common code.
                     19DC  5397
                     19DC  5398 SET_REGION_DATA:
00002258'EF    3F    19DC  5399          PUSHAW  LAST_DATA_LINE          ; Set the scrolling region to use only
00002254'EF    3F    19E2  5400          PUSHAW  FIRST_DATA_LINE         ;  the data area of the display.
          2C   11    19E8  5401          BRB     10$                     ; Branch to common code.
                     19EA  5402
                     19EA  5403 SET_REGION_COMMAND:
00002248'EF    0000225C'EF  D0  19EA  5404          MOVL    CMD_LINE,LINE_NUM       ; Set the scrolling region to the last
00002260'EF    3F    19F5  5405          PUSHAW  LAST_LINE               ;  two lines of the display.
00002248'EF    3F    19FB  5406          PUSHAW  LINE_NUM                ;
          13   11    1A01  5407          BRB     10$                     ; Branch to common code.
                     1A03  5408
                     1A03  5409 SET_REGION_HELP:
00002248'EF    07    D0  1A03  5410          MOVL    #HEADER_END,LINE_NUM    ; Set the scrolling region to the area
00002260'EF    3F    1A0A  5411          PUSHAW  LAST_LINE               ;  from the line following the header
00002248'EF    3F    1A10  5412          PUSHAW  LINE_NUM                ;  to the last line of the display.
                     1A16  5413
```

```
                00000000'GF   02   FB  1A16  5414 10$:    CALLS   #2,G^LIB$SET_SCROLL     ;
                              05  1A1D  5415              RSB                             ;
                                  1A1E  5416
                                  1A1E  5417              .DSABL  LSB
                                  1A1E  5418
```

```
                                K 11

                        .SBTTL  AST routines.
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is entered when data has been placed in the mailbox
                ;       associated with the terminal.  This data may be from a broadcast
                ;       message, unsolicited input from the terminal, or terminal hangup.
                ;       This routine simply sets an event flag and return so that we do
                ;       not remain at AST delivery very long.
                ;
                ; OUPUTS:
                ;
                ;       AST event flag set.
                ;
                ;--

0000            .ENTRY  AST_DEL,^M<>
                        $SETEF_S -                              ; Set the AST event flag.
                                EFN=#EVF$V_AST_FLAG             ;
  04                    RET
```

```
                     1A2A  5444 ;++
                     1A2A  5445 ;
                     1A2A  5446 ; FUNCTIONAL DESCRIPTION:
                     1A2A  5447 ;
                     1A2A  5448 ;       This routine is entered when the user types control-C on the terminal.
                     1A2A  5449 ;       It simply sets the exit flag to indicate that the user has requested
                     1A2A  5450 ;       that the program exit and then transfers control to the exit routine.
                     1A2A  5451 ;
                     1A2A  5452 ; OUTPUTS:
                     1A2A  5453 ;
                     1A2A  5454 ;       Exit flag set at MISC_FLAGS.
                     1A2A  5455 ;
                     1A2A  5456 ;--
                     1A2A  5457
                0000 1A2A  5458 .ENTRY  CNTRLC_AST_DEL,^M<>
                     1A2C  5459
     50  01     D0  1A2C  5460         MOVL    #SS$_NORMAL,R0          ; Indicate normal exit.
                     1A2F  5461         $EXIT_S                         ; Exit.
                     1A38  5462         RET
                     1A39  5463
                     1A39  5464
```

```
                     1A39  5466 ;++
                     1A39  5467 ;
                     1A39  5468 ; FUNCTIONAL DESCRIPTION:
                     1A39  5469 ;
                     1A39  5470 ;       This routine queues a read over the mailbox channel which has been
                     1A39  5471 ;       associated with the terminal and then enables tne mailbox.
                     1A39  5472 ;
                     1A39  5473 ; CALLING SEQUENCE:
                     1A39  5474 ;
                     1A39  5475 ;       JSB     SET_AST
                     1A39  5476 ;
                     1A39  5477 ; CALLED BY:
                     1A39  5478 ;
                     1A39  5479 ;       INIT
                     1A39  5480 ;       MAIN_CONT
                     1A39  5481 ;
                     1A39  5482 ; RETURN STATUS:
                     1A39  5483 ;
                     1A39  5484 ;       SS$_NORMAL returned in R0.
                     1A39  5485 ;--
                     1A39  5486
                     1A39  5487 SET_AST:
                     1A39  5488         $QIO_S                          ; Set the AST.
                     1A39  5489                 CHAN=MBXCHAN,-
                     1A39  5490                 FUNC=#IO$_READVBLK,-
                     1A39  5491                 IOSB=IOSB,-
                     1A39  5492                 ASTADR=AST_DEL,-
                     1A39  5493                 P1=MBX_BUF,-
                     1A39  5494                 P2=#MBX_BUF_LEN
  2C 50    E9        1A69  5495         BLBC    R0,99$                  ; Branch on error.
                     1A6C  5496
                     1A6C  5497         $QIO_S -                        ; Enable mailbox.
                     1A6C  5498                 CHAN=IN_CHAN,-
                     1A6C  5499                 FUNC=#IO$_WRITEVBLK!IO$M_ENABLMBX,-
                     1A6C  5500                 IOSB=IOSB,-
                     1A6C  5501                 P1=CMD_BUF,-
                     1A6C  5502                 P2=#0
                     1A95  5503
  50    01    D0     1A95  5504         MOVL    #SS$_NORMAL,R0
                05     1A98  5505 99$:    RSB
                     1A99  5506
                     1A99  5507
                     1A99  5508         .ENABL  LSB
                     1A99  5509
```

```
1A99  5511 ;++
1A99  5512 ;
1A99  5513 ; FUNCTIONAL DESCRIPTION:
1A99  5514 ;
1A99  5515 ;       This routine is used to erase the screen from the end of the report up
1A99  5516 ;       to the last two lines of the display.  It also erases the line where
1A99  5517 ;       the scrolling signals appear if they are being output.  This insures
1A99  5518 ;       that the scrolling signals always reflect the correct status and also
1A99  5519 ;       produces a blinking effect.
1A99  5520 ;
1A99  5521 ;       This routine also determines if the bottom two lines of the display
1A99  5522 ;       should be scrolled.  This scrolling is done only if there is additional
1A99  5523 ;       information on the bottom command line or if there is no information
1A99  5524 ;       on the bottom command line and the top command line does not contain
1A99  5525 ;       a broadcast message. The results (listed in priority) appear as follows:
1A99  5526 ;
1A99  5527 ;       1) Command input and error messages are scrolled in the command region
1A99  5528 ;          on each refresh until they disappear from the display.
1A99  5529 ;
1A99  5530 ;       2) Broadcast messages are scrolled in the command region until the last
1A99  5531 ;          broadcast message is on the top line in the command region at which
1A99  5532 ;          point it is left there until the user takes action to remove it.
1A99  5533 ;
1A99  5534 ;       3) Scrolling signals are erased and are restored after the next update
1A99  5535 ;          on the first blank line in the command region.
1A99  5536 ;
1A99  5537 ; CALLING SEQUENCE:
1A99  5538 ;
1A99  5539 ;       JSB     ERASE_PAGE
1A99  5540 ;
1A99  5541 ; CALLED BY:
1A99  5542 ;
1A99  5543 ;       DISPLAY_END
1A99  5544 ;
1A99  5545 ; INPUTS:
1A99  5546 ;
1A99  5547 ;       Number of the line following the report is stored in LINE_NUM.
1A99  5548 ;
1A99  5549 ; IMPLICIT INPUTS:
1A99  5550 ;
1A99  5551 ;       SCROLL_FLAGS    have been set to indicate the direction the screen can
1A99  5552 ;                       be scrolled.
1A99  5553 ;       TOP_CMD_LINE and BOT_CMD_LINE indicate the type of data currently being
1A99  5554 ;               displayed in the command scroll region.
1A99  5555 ;
1A99  5556 ; IMPLICIT OUTPUTS:
1A99  5557 ;
1A99  5558 ;       Bottom two lines of display are scrolled.
1A99  5559 ;       TOP_CMD_LINE and BOT_CMD_LINE are updated.
1A99  5560 ;
1A99  5561 ; RETURN STATUS:
1A99  5562 ;
1A99  5563 ;       SS$_NORMAL or failed status returned in R0.
1A99  5564 ;
1A99  5565 ; REGISTER USAGE:
1A99  5566 ;
1A99  5567 ;       R2 -    Used to determine when all the lines have been erased.
```

```
                                1A99  5568 ;--
                                1A99  5569
                                1A99  5570 ERASE_PAGE:
                     50   01    D0  1A99  5571          MOVL    #SS$_NORMAL,R0                  ; Assume success.
OD 00000476'EF       03   E0    1A9C  5572          BBS     #MFS$V_RIP,MISC_FLAGS,5$        ; Branch if refresh in progress.
          00000428'EF       D1  1AA4  5573          CMPL    NUM_LINES,-                     ; Do we have the same number of lines
          0000042C'EF           1AAA  5574                  PRIOR_NUM_LINES                 ;  of data as last pass?
                       1F   13  1AAF  5575          BEQL    14$                             ; If so skip the erase.
                       04   BB  1AB1  5576 5$:      PUSHR   #^M<R2>
     52   00002258'EF       D0  1AB3  5577          MOVL    LAST_DATA_LINE,R2               ; Erase all the remaining data.
     52   00002248'EF       D1  1ABA  5578 10$:     CMPL    LINE_NUM,R2                     ; Any lines to erase?
                       0B   14  1AC1  5579          BGTR    11$                             ; Branch if not.
                     093C   30  1AC3  5580          BSBW    ERASE_LINE                      ; Erase one line.
                            1AC6  5581          JLBC    R0,99$                          ; Branch on error.
                       EC   11  1ACC  5582          BRB     10$                             ; Loop.
                       04   BA  1ACE  5583 11$:     POPR    #^M<R2>
                            1AD0  5584
                            1AD0  5585 ;
                            1AD0  5586 ; If we are outputing "more" on the bottom line then erase the line so that it
                            1AD0  5587 ; reflects the correct status.
                            1AD0  5588 ;
                            1AD0  5589
          00000428'EF       D0  1AD0  5590 14$:     MOVL    NUM_LINES,-                     ; Save number of lines for next pass.
          0000042C'EF           1AD6  5591                  PRIOR_NUM_LINES                 ;
     00000428'EF       07   C3  1ADB  5592          SUBL3   #7,NUM_LINES,-                  ; Save number of times we can up scroll.
          00000430'EF           1AE2  5593                  SCROLL_LINES                    ;
                       00   E1  1AE7  5594          BBC     #CLS$V_CLUB,-                   ; Branch if not displaying the CLUSTER
        07 00000001'EF           1AE9  5595                  CLASS_SELECT,15$                ;  class.
     00000430'EF       07   C2  1AEF  5596          SUBL    #7,SCROLL_LINES                 ; Otherwise, adjust number of lines that
                            1AF6  5597                                                      ;  can be scrolled.
                       04   E1  1AF6  5598 15$:     BBC     #SCR$V_MORE,-                   ; Branch if none of the scroll flags
                 2E 0000047A'EF     1AF8  5599                  SCROLL_FLAGS,20$                ;  have been set.
                          04   00  EC  1AFE  5600          CMPV    #SCR$V_MORE_RT,#4,-             ; Are any of the scroll flags still
                00   0000047A'EF     1B01  5601                  SCROLL_FLAGS,#0                 ;  set?
                       07   12  1B07  5602          BNEQ    16$                             ; Branch if so.
                0000047A'EF   10   8A  1B09  5603          BICB    #SCR$M_MORE,SCROLL_FLAGS        ; Otherwise clear the summary flag.
          00000471'EF       95  1B10  5604 16$:     TSTB    BOT_CMD_LINE                    ; Is the bottom command line blank?
                       14   12  1B16  5605          BNEQ    20$                             ; Branch if not.
00002248'EF   00002260'EF       D0  1B18  5606          MOVL    LAST_LINE,LINE_NUM              ; Get the number of the bottom command
                            1B23  5607                                                      ;  line.
                     08DC   30  1B23  5608          BSBW    ERASE_LINE                      ; Erase the line that we are outputing
                            1B26  5609                                                      ;  "more" on.
                            1B26  5610          JLBC    R0,99$                          ; Branch on error.
          00000458'EF       D5  1B2C  5611 20$:     TSTL    SCROLL_DELAY                    ; Time to scroll the bottom two lines?
                       09   13  1B32  5612          BEQL    30$                             ; Branch if so.
          00000458'EF       D7  1B34  5613          DECL    SCROLL_DELAY                    ; Count down the intervals before it's
                            1B3A  5614                                                      ;  time to scroll.
                     0108   31  1B3A  5615          BRW     99$                             ; Done for now.
          00000471'EF       95  1B3D  5616 30$:     TSTB    BOT_CMD_LINE                    ; Any information on the bottom command
                            1B43  5617                                                      ;  line?
                            1B43  5618          JNEQ    SCROLL_TWO                      ; Branch if so.
          00000470'EF       95  1B48  5619          TSTB    TOP_CMD_LINE                    ; Any information on the top command
                            1B4E  5620                                                      ;  line?
                            1B4E  5621          JEQL    99$                             ; If not, no need to scroll.
                       01   93  1B53  5622          BITB    #CL$M_BRDCST,-                  ; Is there a broadcast message on the
          00000470'EF           1B55  5623                  TOP_CMD_LINE                    ;  top command line?
                            1B5A  5624          JNEQ    99$                             ; If so, don't scroll.
```

```
                                1B5F  5625
                                1B5F  5626 SCROLL_TWO:
        3E 00002247'EF    01    E1  1B5F  5627          BBC     #DVT$V_DCRT,OUT_DEV,40$ ; Branch if terminal is a VT52 type.
                     FE7F CF    16  1B67  5628          JSB     SET_REGION_COMMAND      ; Establish a scrolling region.
                                1B6B  5629          JLBC    R0,99$                  ; Branch on error.
00002248'EF    00002260'EF    D0  1B71  5630          MOVL    LAST_LINE,LINE_NUM      ; Set the cursor to the last line.
                  0000224C'EF  3F  1B7C  5631          PUSHAW  COL_NUM                 ;
                  00002248'EF  3F  1B82  5632          PUSHAW  LINE_NUM                ;
              00000000'GF    02  FB  1B88  5633          CALLS   #2,G^LIB$SET_CURSOR     ;
                                1B8F  5634          JLBC    R0,99$                  ; Branch on error.
              00000000'GF    00  FB  1B95  5635          CALLS   #0,G^LIB$UP_SCROLL      ; Scroll up one line.
                                1B9C  5636          JLBC    R0,99$                  ; Branch on error.
                       008F  31  1BA2  5637          BRW     80$                     ; Jump to common code.
        03 00002247'EF    02  E1  1BA5  5638 40$:     BBC     #DVT$V_HARD,OUT_DEV,43$ ; Branch if output device isn't
                                1BAD  5639                                          ;  hardcopy.
                       0084  31  1BAD  5640          BRW     80$                     ; Otherwise, jump to common code.
              00002247'EF    03  E0  1BB0  5641 43$:     BBS     #DVT$V_UNKNOWN,OUT_DEV,-; Branch if terminal type is unknown.
                           7C      1BB7  5642                  80$                     ;
                  00000471'EF  95  1BB8  5643          TSTB    BOT_CMD_LINE            ; Any information on the bottom command
                                1BBE  5644                                          ;  line?
                           4B  13  1BBE  5645          BEQL    50$                     ; Branch if not.
00002248'EF    0000225C'EF    D0  1BC0  5646          MOVL    CMD_LINE,LINE_NUM       ; Set the cursor to the top command
                  0000224C'EF  3F  1BCB  5647          PUSHAW  COL_NUM                 ;  line.
                  00002248'EF  3F  1BD1  5648          PUSHAW  LINE_NUM                ;
              00000000'GF    02  FB  1BD7  5649          CALLS   #2,G^LIB$SET_CURSOR     ;
                        64 50    E9  1BDE  5650          BLBC    R0,99$                  ; Branch on error.
                           01  93  1BE1  5651          BITB    #CL$M_BRDCST,-          ; Is there a broadcast message on the
                  00000471'EF      1BE3  5652                  BOT_CMD_LINE            ;  on the bottom command line.
                           12  13  1BE8  5653          BEQL    45$                     ; If not, must be error message.
                  00001DF2'EF  3F  1BEA  5654          PUSHAW  LAST_BRDCST_DSCR        ; Output the last broadcast message on
              00000000'GF    01  FB  1BF0  5655          CALLS   #1,G^LIB$PUT_SCREEN     ;  the top command line.
                        4B 50    E9  1BF7  5656          BLBC    R0,99$                  ; Branch on error.
                           03  11  1BFA  5657          BRB     46$                     ; Jump to common code.
                         F95E  30  1BFC  5658 45$:     BSBW    SIGNAL                  ; Output the last error message on the
                                1BFF  5659                                          ;  top command line.
                00000000'GF    00  FB  1BFF  5660 46$:     CALLS   #0,G^LIB$ERASE_PAGE     ; Erase the bottom line.
                        3C 50    E9  1C06  5661          BLBC    R0,99$                  ; Branch on error.
                           29  11  1C09  5662          BRB     80$                     ; Jump to common code.
                  00000470'EF  95  1C0B  5663 50$:     TSTB    TOP_CMD_LINE            ; Any information on the top command
                                1C11  5664                                          ;  line?
                           32  13  1C11  5665          BEQL    99$                     ; Branch if not.
00002248'EF    0000225C'EF    D0  1C13  5666          MOVL    CMD_LINE,LINE_NUM       ; Erase the top command line.
                  0000224C'EF  3F  1C1E  5667          PUSHAW  COL_NUM                 ;
                  00002248'EF  3F  1C24  5668          PUSHAW  LINE_NUM                ;
              00000000'GF    02  FB  1C2A  5669          CALLS   #2,G^LIB$ERASE_PAGE     ;
                        11 50    E9  1C31  5670          BLBC    R0,99$                  ; Branch on error.
                  00000471'EF  90  1C34  5671 80$:     MOVB    BOT_CMD_LINE,-          ; Bottom command line information now
                  00000470'EF      1C3A  5672                  TOP_CMD_LINE            ;  on the command line.
                  00000471'EF  94  1C3F  5673          CLRB    BOT_CMD_LINE            ; No information on bottom command
                                1C45  5674                                          ;  line.
                           05  1C45  5675 99$:     RSB
                                1C46  5676
                                1C46  5677          .DSABL  LSB
                                1C46  5678
                                1C46  5679
```

```
                        1C46  5681          .SBTTL  Display header generation.
                        1C46  5682 ;++
                        1C46  5683 ;
                        1C46  5684 ; FUNCTIONAL DESCRIPTION:
                        1C46  5685 ;
                        1C46  5686 ;       This routine combines the class titles and the field titles to form the
                        1C46  5687 ;       display header.  It also adds the necessary strings to draw the box
                        1C46  5688 ;       around the header.  It outputs a message identifying the report and the
                        1C46  5689 ;       displays the time in the upper right hand corner.
                        1C46  5690 ;
                        1C46  5691 ; CALLING SEQUENCE:
                        1C46  5692 ;
                        1C46  5693 ;       JSB     GET_HEADER
                        1C46  5694 ;
                        1C46  5695 ;
                        1C46  5696 ; CALLED BY:
                        1C46  5697 ;
                        1C46  5698 ;       DISPLAY_CLUB
                        1C46  5699 ;       DISPLAY_SCS
                        1C46  5700 ;       DISPLAY_PTP
                        1C46  5701 ;
                        1C46  5702 ; OUTPUTS:
                        1C46  5703 ;
                        1C46  5704 ;         The entire header is stored in the screen buffer for
                        1C46  5705 ;       output using LIB$PUT_BUFFER.
                        1C46  5706 ;
                        1C46  5707 ; RETURN STATUS:
                        1C46  5708 ;
                        1C46  5709 ;       R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                        1C46  5710 ;
                        1C46  5711 ; REGISTER USAGE:
                        1C46  5712 ;
                        1C46  5713 ;       R7  -   Used to in formating the system id.
                        1C46  5714 ;       R10 -   Used to in formating the system id.
                        1C46  5715 ;       R11 -   Pointer to descriptor for data to be stored in the screen
                        1C46  5716 ;               buffer.
                        1C46  5717 ;
                        1C46  5718 ;--
                        1C46  5719
                        1C46  5720 GET_HEADER:
              0800 8F   BB  1C46  5721          PUSHR   #^M<R11>
  2B 00000476'EF  03    E0  1C4A  5722          BBS     #MF$V_RIP,MISC_FLAGS,30$; Branch if refresh in progress.
                        1C52  5723 ;
                        1C52  5724 ; When not doing a refresh, just output the time in the upper right hand corner.
                        1C52  5725 ;
                        1C52  5726 ; If the CLUSTER class is being display along with any SCS class, then display
                        1C52  5727 ; the time when GET_HEADER is called for the CLUSTER class only, so that we
                        1C52  5728 ; don't update the time twice.
                        1C52  5729 ;
                        1C52  5730
                  00    E1  1C52  5731          BBC     #RPT$V_CLSTR,-          ; Branch if displaying the local report.
    10 00000000'EF          1C54  5732                  CURRENT_REPORT,10$      ;
                  00    E0  1C5A  5733          BBS     #CLS$V_CLUB,-           ; Branch if this header is for the
    08 00000001'EF          1C5C  5734                  CLASS_SELECT,10$        ;  CLUSTER class.
                  00    E0  1C62  5735          BBS     #CLS$V_CLUB,-           ; Branch if we have already output
    09 000021FC'EF          1C64  5736                  SAVED_CLASSES,20$       ;  the time.
                01F5    30  1C6A  5737 10$:     BSBW    OUTPUT_TIME             ; Output the time.
```

```
                00002248'EF   D6  1C6D  5738          INCL    LINE_NUM
            00002248'EF   05   C0  1C73  5739 20$:     ADDL    #5,LINE_NUM             ; Skip past header info.
                        0112   31  1C7A  5740          BRW     998$                    ; And we're done.
                               1C7D  5741
                        04C4   30  1C7D  5742 30$:     BSBW    GET_TITLES              ; Build the title line in TITLE_BUF.
                               1C80  5743          JLBC    R0,999$                 ; Branch on error.
                01   000021C4'EF   D1  1C86  5744          CMPL    HEADER_SIZE,#1          ; Are there any classes or fields being
                               1C8D  5745                                              ;  displayed?
                          06   12  1C8D  5746          BNEQ    40$                     ; Branch if so.
                000021C4'EF   D4  1C8F  5747          CLRL    HEADER_SIZE             ; Otherwise, make sure the first field
                               1C95  5748                                              ;  delimiter doesn't get displayed.
                        0608   30  1C95  5749 40$:     BSBW    GET_CLASSES             ; Build the class line in CLASS_BUF.
                               1C98  5750          JLBC    R0,999$                 ; Branch on error.
             05 00002247'EF   01   E1  1C9E  5751          BBC     #DVT$V_DCRT,OUT_DEV,50$ ; Branch if terminal isn't a DECcrt.
                        02F4   30  1CA6  5752          BSBW    BUILD_DECCRT_BOX        ; Get the four lines of data that make
                               1CA9  5753                                              ;  up the boxes around the header.
                          03   11  1CA9  5754          BRB     60$                     ; Jump to common code.
                        0296   30  1CAB  5755 50$:     BSBW    BUILD_VT52_BOX          ; Build the line for above the classes
                               1CAE  5756                                              ;  in CLASS_BOX_BUF and the line for
                               1CAE  5757                                              ;  above and below the field titles in
                               1CAE  5758                                              ;  FIELD_BOX_BUF.
                               1CAE  5759
                          00   E1  1CAE  5760 60$:     BBC     #RPT$V_CLSTR,-          ; Branch if displaying the local report.
                7B 00000000'EF       1CB2  5761          CURRENT_REPORT,90$      ;
                          00   E1  1CB6  5762          BBC     #CLS$V_CLUB,-           ; Branch if this header isn't for the
                31 00000001'EF       1CB8  5763          CLASS_SELECT,70$        ;  CLUSTER class.
                   000021C4'EF   D0  1CBE  5764          MOVL    HEADER_SIZE,-           ; Save size of CLUB header.
                   000021C8'EF       1CC4  5765          CLUB_HEADER_SIZE        ;
                   00000425'EF   90  1CC9  5766          MOVB    NUM_FIELDS,-            ; Save the number of CLUB fields being
                   00000427'EF       1CCF  5767          CLUB_NUM_FIELDS         ;  displayed.
                          00C0   30  1CD4  5768          BSBW    OUTPUT_DDM              ; Output the display descriptor message.
                   0000007E 8F   D3  1CD7  5769          BITL    #CLS$M_ANY_SCS,-        ; Are any SCS fields being displayed in
                   000021FC'EF       1CDD  5770          SAVED_CLASSES           ;  this report?
                          03   12  1CE2  5771          BNEQ    65$                     ; If so, don't output the time yet.
                        017B   30  1CE4  5772          BSBW    OUTPUT_TIME             ; Otherwise, do it now.
                00002248'EF   D6  1CE7  5773 65$:     INCL    LINE_NUM                ; Update line number.
                          4E   11  1CED  5774          BRB     100$                    ; Branch to common code.
                000021C4'EF   D0  1CEF  5775 70$:     MOVL    HEADER_SIZE,-           ; Save size of SCS header.
                000021CC'EF       1CF5  5776          SCS_HEADER_SIZE         ;
                00000425'EF   90  1CFA  5777          MOVB    NUM_FIELDS,-            ; Save the number of SCS fields being
                00000426'EF       1D00  5778          SCS_NUM_FIELDS          ;  displayed.
                          00   E1  1D05  5779 80$:     BBC     #CLS$V_CLUB,-           ; If CLUB bit is clear we need to output
                24 000021FC'EF       1D07  5780          SAVED_CLASSES,90$       ;  the display descriptor.
                00000426'EF   91  1D0D  5781          CMPB    SCS_NUM_FIELDS,-        ; Are there more SCS fields than CLUB
                00000427'EF       1D13  5782          CLUB_NUM_FIELDS         ;  fields being displayed?
                          0B   18  1D18  5783          BGEQ    85$                     ; Branch if so.
                00000427'EF   90  1D1A  5784          MOVB    CLUB_NUM_FIELDS,-       ; Otherwise, limit left-scrolling by the
                00000425'EF       1D20  5785          NUM_FIELDS              ;  number of CLUB fields.
                        013A   30  1D25  5786 85$:     BSBW    OUTPUT_TIME             ; Output the time.
            00002248'EF   08   C0  1D28  5787          ADDL    #8,LINE_NUM             ; Skip past the CLUB display area.
                          0C   11  1D2F  5788          BRB     100$                    ; Branch to common code.
                        0063   30  1D31  5789 90$:     BSBW    OUTPUT_DDM              ; Output the display descriptor message.
                        012B   30  1D34  5790          BSBW    OUTPUT_TIME             ; Output the time.
                00002248'EF   D6  1D37  5791          INCL    LINE_NUM                ; Update line number.
                               1D3D  5792
             5B   00000763'EF   DE  1D3D  5793 100$:    MOVAL   CLASS_BOX_DSCR,R11      ; Point to descriptor for top of box.
                        071F   30  1D44  5794          BSBW    DRAW_LINE               ; Output top of box.
```

```
                48 50   E9  1D47  5795           BLBC    R0,999$                         ; Branch on error.
          5B    00001593'EF   DE  1D4A  5796           MOVAL   CLASS_DSCR,R11                  ; Point to descriptor for the classes.
                  0712  30  1D51  5797           BSBW    DRAW_LINE                       ; Output class names.
                3B 50   E9  1D54  5798           BLBC    R0,999$                         ; Branch on error.
          5B    0000096B'EF   DE  1D57  5799           MOVAL   FIELD_TOP_BOX_DSCR,R11          ; Point to descriptor for line above
                                1D5E  5800                                                   ;  the field names.
                  0705  30  1D5E  5801           BSBW    DRAW_LINE                       ; Output the line.
                2E 50   E9  1D61  5802           BLBC    R0,999$                         ; Branch on error.
          5B    0000118B'EF   DE  1D64  5803           MOVAL   TITLE_DSCR,R11                  ; Point to descriptor for the field
                                1D6B  5804                                                   ;  titles.
                  06F8  30  1D6B  5805           BSBW    DRAW_LINE                       ; Output the field names.
                21 50   E9  1D6E  5806           BLBC    R0,999$                         ; Branch on error.
       09 00002247'EF   01  E1  1D71  5807           BBC     #DVT$V_DCRT,OUT_DEV,110$        ; Branch if terminal is not a DECcrt.
          5B    00000B73'EF   DE  1D79  5808           MOVAL   FIELD_BOT_BOX_DSCR,R11          ; Point to descriptor for line below
                                1D80  5809                                                   ;  the field names.
                    07  11  1D80  5810           BRB     120$                            ; Jump to common code.
          5B    0000096B'EF   DE  1D82  5811 110$:   MOVAL   FIELD_TOP_BOX_DSCR,R11          ; Point to descriptor for line above
                                1D89  5812                                                   ;  the field names.
                  06DA  30  1D89  5813 120$:   BSBW    DRAW_LINE                       ; Complete the box around the field
                03 50   E9  1D8C  5814           BLBC    R0,999$                         ; Branch on error.
                                1D8F  5815                                                   ;  names.
              50    01  D0  1D8F  5816 998$:   MOVL    #SS$_NORMAL,R0
              0800 8F   BA  1D92  5817 999$:   POPR    #^M<R11>
                        05  1D96  5818           RSB
                            1D97  5819
                            1D97  5820
```

```
                     1D97  5822          .SBTTL  -       Output the display description
                     1D97  5823 ;++
                     1D97  5824 ;
                     1D97  5825 ; FUNCTIONAL DESCRIPTION:
                     1D97  5826 ;
                     1D97  5827 ;       This routine is called by GET_HEADER only if we are doing a refresh
                     1D97  5828 ;       pass.  It outputs the description of the display which appears on the
                     1D97  5829 ;       top line of the report.
                     1D97  5830 ;
                     1D97  5831 ; CALLING SEQUENCE:
                     1D97  5832 ;
                     1D97  5833 ;       JSB     OUTPUT_DDM
                     1D97  5834 ;
                     1D97  5835 ;
                     1D97  5836 ; CALLED BY:
                     1D97  5837 ;
                     1D97  5838 ;       GET_HEADER
                     1D97  5839 ;
                     1D97  5840 ; OUTPUTS:
                     1D97  5841 ;
                     1D97  5842 ;       Display discription message is output to the screen buffer.
                     1D97  5843 ;       Length of message is stored in DDM_SIZE.
                     1D97  5844 ;
                     1D97  5845 ; IMPLICIT OUTPUTS:
                     1D97  5846 ;
                     1D97  5847 ;       LINE_NUM is set to 1.
                     1D97  5848 ;
                     1D97  5849 ; RETURN STATUS:
                     1D97  5850 ;
                     1D97  5851 ;       R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                     1D97  5852 ;
                     1D97  5853
                     1D97  5854 OUTPUT_DDM:
         048C 8F     BB  1D97  5855          PUSHR   #^M<R2,R3,R7,R10>
              53     D4  1D9B  5856          CLRL    R3                      ; Initialize string length.
00002248'EF  01     D0  1D9D  5857          MOVL    #1,LINE_NUM             ; Output description on line one.
     0000224C'EF    3F  1DA4  5858          PUSHAW  COL_NUM                 ; Setup for LIB$PUT_SCREEN.
     00002248'EF    3F  1DAA  5859          PUSHAW  LINE_NUM                ;
              00    E1  1DB0  5860          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
  09 00000000'EF        1DB2  5861                  CURRENT_REPORT,10$      ;  cluster report.
52   0000018A'EF    DE  1DB8  5862          MOVAL   CLUSTER_MSG,R2          ; Get address of message
                         1DBF  5863                                          ;  "View of Cluster from" string.
              07    11  1DBF  5864          BRB     20$                     ; Jump to common code.
52   000001A6'EF    DE  1DC1  5865 10$:     MOVAL   LOCAL_MSG,R2            ; Get address of "Local ports on".
          53    62    A0  1DC8  5866 20$:     ADDW    (R2),R3                 ; Calculate string length thus far.
                52    DD  1DCB  5867          PUSHL   R2                      ; Output the message.
00000000'GF    03    FB  1DCD  5868          CALLS   #3,G^LIB$PUT_SCREEN     ;
                         1DD4  5869          JLBC    R0,99$                  ; Branch on error.
53   000001BC'EF    A0  1DDA  5870          ADDW    MSG1_DSCR,R3            ; Add length of system id message to
                         1DE1  5871                                          ;  string length.
     000001BC'EF    DF  1DE1  5872          PUSHAL  MSG1_DSCR               ; Push address of " system ID# "
                         1DE7  5873                                          ;  string.
00000000'GF    01    FB  1DE7  5874          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the message.
                         1DEE  5875          JLBC    R0,99$                  ; Branch on error.
5A   00000000'EF    DE  1DF4  5876          MOVAL   LOCSYSID_DSCR,R10       ; Point to descriptor for the local
                         1DFB  5877                                          ;  system ID.
      57    10 AA    D0  1DFB  5878          MOVL    FD$L_OFFSET(R10),R7     ; Get the address of the data.
```

```
                077D  30  1DFF  5879          BSBW    FORMAT_DATA             ; Format the data.
                      1E02  5880          JLBC    R0,99$                  ; Branch on error.
53  00002153'EF   A0  1E08  5881          ADDW    FORMAT_BUF_DSCR,R3      ; Add length of system id to string
                      1E0F  5882                                          ;  length.
    00002153'EF   DF  1E0F  5883          PUSHAL  FORMAT_BUF_DSCR         ; Push the address of the formatted
                      1E15  5884                                          ;  data.
00000000'GF   01  FB  1E15  5885          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the system ID number.
                      1E1C  5886          JLBC    R0,99$                  ; Branch on error.
53  000001CF'EF   A0  1E22  5887          ADDW    MSG2_DSCR,R3            ; Add length of node message to string
                      1E29  5888                                          ;  length.
    000001CF'EF   DF  1E29  5889          PUSHAL  MSG2_DSCR               ; Push the address of "  node: "
                      1E2F  5890                                          ;  string.
00000000'GF   01  FB  1E2F  5891          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the message.
                      1E36  5892          JLBC    R0,99$                  ; Branch on error.
53  00000448'EF   A0  1E3C  5893          ADDW    NODE_NAME_DSCR,R3       ; Add length of node name to string
                      1E43  5894                                          ;  length.
    00000448'EF   DF  1E43  5895          PUSHAL  NODE_NAME_DSCR          ; Push address of local nodename.
00000000'GF   01  FB  1E49  5896          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the nodename.
                      1E50  5897          JLBC    R0,99$                  ; Branch on error.
000020D1'EF   53  B0  1E56  5898          MOVW    R3,DDM_SIZE             ; Save size of display descriptor
                      1E5D  5899                                          ;  message.
          048C 8F   BA  1E5D  5900 99$:     POPR    #^M<R2,R3,R7,R10>
                      05  1E61  5901          RSB
                          1E62  5902
                          1E62  5903
```

```
                                                                I 12
                                        .SBTTL  -       Output the time
                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine is called by GET_HEADER to output the time on the
                                ;       top line of the display.
                                ;
                                ;       If the size of the either the CLUSTER class header, the SCS class
                                ;       header in the CLUSTER report, or the header for the LOCAL report
                                ;       (depending on what's being displayed) is larger that the display
                                ;       description message + 10, then the time is output along the right
                                ;       hand border of the header.  If it's smaller, then the time is
                                ;       output at the extreme right.  This is done to improve the appearance
                                ;       of the display and to allow hardcopy output on 80 column paper without
                                ;       cutting off the time.
                                ;
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     OUTPUT_TIME
                                ;
                                ; CALLED BY:
                                ;
                                ;       GET_HEADER
                                ;
                                ; INPUTS:
                                ;
                                ;       DDM_SIZE contains the length of the display description message.
                                ;       SCREEN_WIDTH contains the current width of the screen.
                                ;       HEADER_SIZE, CLUB_HEADER_SIZE and SCS_HEADER_SIZE contain the
                                ;       width of the header for the classes being displayed.
                                ;
                                ; OUTPUTS:
                                ;
                                ;       Time is output to the screen buffer.
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       LINE_NUM is set to 1.
                                ;
                                ; RETURN STATUS:
                                ;
                                ;       R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                                ;
                                
                                OUTPUT_TIME:
                         3C BB  1E62  5952              PUSHR   #^M<R2,R3,R4,R5>
          00002248'EF    01 D0  1E64  5953              MOVL    #1,LINE_NUM             ;
00 0224C'EF 000020D1'EF  01 A1  1E6B  5954              ADDW3   #1,DDM_SIZE,COL_NUM     ; Get number of column after display
                                1E77  5955                                              ;  description message.
          00002248'EF    01 B0  1E77  5956              MOVW    #1,LINE_NUM             ; Time goes on line one.
                         00 E1  1E7E  5957              BBC     #RPT$V_CLSTR,-          ; Branch if not displaying the cluster
                                1E80  5958                      CURRENT_REPORT,10$      ;  report.
                            D1  1E86  5959              CMPL    SCS_HEADER_SIZE,-       ; Use the larger header to determine
                                1E8C  5960                      CLUB_HEADER_SIZE        ;  where the time should be output.
                         09 14  1E91  5961              BGTR    5$                      ; Branch if SCS is larger.
```

```
                54   000021C8'EF   D0  1E93  5962          MOVL    CLUB_HEADER_SIZE,R4     ; Get the header size of the CLUB.
                                10   11  1E9A  5963          BRB     20$                     ; Branch to common code.
                54   000021CC'EF   D0  1E9C  5964 5$:      MOVL    SCS_HEADER_SIZE,R4      ; Get the header size of the SCS.
                                07   11  1EA3  5965          BRB     20$                     ; Branch to common code.
                54   000021C4'EF   D0  1EA5  5966 10$:     MOVL    HEADER_SIZE,R4          ; Get the header size for the local
                                          1EAC  5967                                          ;  report.
                                          1EAC  5968 ;
                                          1EAC  5969 ; Determine how many spaces to output between the display discription message
                                          1EAC  5970 ; and the time based on the size on the header.  If it's less than 10, then
                                          1EAC  5971 ; output the time at the extreme right.
                                          1EAC  5972 ;
                                          1EAC  5973
52   000020D1'EF   000020B5'EF   A1  1EAC  5974 20$:     ADDW3   TIME_DSCR,DDM_SIZE,R2   ; Get size of message + time.
                     53   54   52   A3  1EB8  5975          SUBW3   R2,R4,R3                ; Get the number of columns to erase.
                          06   53   B1  1EBC  5976          CMPW    R3,#6                   ; Less than 6 blanks?
                               18   18  1EBF  5977          BGEQ    40$                     ; Branch if not.
                  0A 00002247'EF    02  E0  1EC1  5978          BBS     #DVT$V_HARD,OUT_DEV,30$ ; Branch if output is hardcopy.
             53   00002222'EF    52   A3  1EC9  5979          SUBW3   R2,SCREEN_WIDTH,R3    ; Get the number of columns to erase.
                               06   11  1ED1  5980          BRB     40$                     ; Branch to common code.
                53   0050 8F    52   A3  1ED3  5981 30$:     SUBW3   R2,#80,R3               ; Get the number of columns to erase.
                               53   D5  1ED9  5982 40$:     TSTL    R3                      ; Skip decrement if fill length is 0.
                               02   13  1EDB  5983          BEQL    50$                     ;
                               53   B7  1EDD  5984          DECW    R3                      ; One less for aesthetics...
                    0000199B'EF    53   B0  1EDF  5985 50$:     MOVW    R3,DATA_DSCR            ; Use data descriptor for fill.
                    0000199F'FF    00   2C  1EE6  5986          MOVC5   #0,@DATA_DSCR+4,-       ; Fill the descriptor with some spaces.
               0000199F'FF    53   20       1EED  5987                  #^A/ /,R3,@DATA_DSCR+4  ;
                         0000224C'EF   3F  1EF4  5988          PUSHAW  COL_NUM                 ; Setup for LIB$PUT_SCREEN.
                         00002248'EF   3F  1EFA  5989          PUSHAW  LINE_NUM                ;
                         0000199B'EF   DF  1F00  5990          PUSHAL  DATA_DSCR               ; Output the spaces.
                    00000000'GF    03   FB  1F06  5991          CALLS   #3,G^LIB$PUT_SCREEN     ;
                                          1F0D  5992          JLBC    R0,99$                  ; Branch on error.
                         000020B5'EF   DF  1F13  5993          PUSHAL  TIME_DSCR               ; Push address to store time.
               00000000'GF    01   FB  1F19  5994          CALLS   #1,G^LIB$DATE_TIME      ; Get the current date and time.
                          17   50   E9  1F20  5995          BLBC    R0,99$                  ; Branch on error.
                         000020B5'EF   DF  1F23  5996          PUSHAL  TIME_DSCR               ;
               00000000'GF    01   FB  1F29  5997          CALLS   #1,G^LIB$PUT_SCREEN     ;
                          07   50   E9  1F30  5998          BLBC    R0,99$                  ; Branch on error.
               00000000'GF    00   FB  1F33  5999          CALLS   #0,G^LIB$ERASE_LINE     ; Erase the rest of the line.
               0000224C'EF    01   D0  1F3A  6000 99$:     MOVL    #1,COL_NUM              ; Restore column number.
                              3C   BA  1F41  6001          POPR    #^M<R2,R3,R4,R5>
                              05   1F43  6002          RSB
                                   1F44  6003
                                   1F44  6004
```

```
                     1F44  6006          .SBTTL          -       Box drawing routines.
                     1F44  6007 ;++
                     1F44  6008 ;
                     1F44  6009 ; FUNCTIONAL DESCRIPTION:
                     1F44  6010 ;
                     1F44  6011 ;       This routine builds the descriptors for the lines which make up the
                     1F44  6012 ;       boxes surrounding the classes and titles in the display header and the
                     1F44  6013 ;       line following the last line of data in the report when the report is
                     1F44  6014 ;       run on a non-VT100 compatable terminal.
                     1F44  6015 ;
                     1F44  6016 ; CALLING SEQUENCE:
                     1F44  6017 ;
                     1F44  6018 ;       JSB     BUILD_VT52_BOX
                     1F44  6019 ;
                     1F44  6020 ; CALLED BY:
                     1F44  6021 ;
                     1F44  6022 ;       GET_HEADER
                     1F44  6023 ;
                     1F44  6024 ; OUTPUTS:
                     1F44  6025 ;       Data for line above and below the titles and the line following the
                     1F44  6026 ;       last line of data stored in FIELD_TOP_BOX_BUF_DSCR.
                     1F44  6027 ;       Data for line above and below the classes stored in CLASS_BOX_BUF_DSCR.
                     1F44  6028 ;
                     1F44  6029 ; RETURN STATUS:
                     1F44  6030 ;
                     1F44  6031 ;       None
                     1F44  6032 ;
                     1F44  6033 ; REGISTER USAGE:
                     1F44  6034 ;
                     1F44  6035 ;       R5 -    Pointer into TITLE_BUF
                     1F44  6036 ;       R6 -    Counter for the number of charcters in the buffers.
                     1F44  6037 ;       R7 -    Pointer into CLASS_BUF
                     1F44  6038 ;       R8 -    Pointer into CLASS_BOX_BUF
                     1F44  6039 ;       R9 -    Pointer into FIELD_TOP_BOX_BUF
                     1F44  6040 ;
                     1F44  6041 ;--
                     1F44  6042
                     1F44  6043 BUILD_VT52_BOX:
         03E0 8F BB  1F44  6044          PUSHR   #^M<R5,R6,R7,R8,R9>
58 0000076B'EF DE    1F48  6045          MOVAL   CLASS_BOX_BUF,R8        ; Get address of buffer for line above
                     1F4F  6046                                          ;  the classes.
59 00000973'EF DE    1F4F  6047          MOVAL   FIELD_TOP_BOX_BUF,R9    ; Get address of buffer for above and
                     1F56  6048                                          ;  below field titles and last line.
56 0000118B'EF 3C    1F56  6049          MOVZWL  TITLE_DSCR,R6           ; Get the number of bytes in TITLE_BUF.
0000096B'EF 56 B0    1F5D  6050          MOVW    R6,FIELD_TOP_BOX_DSCR   ; Save as the number of bytes in
                     1F64  6051                                          ;  FIELD_BOX_BUF.
00000763'EF 56 B0    1F64  6052          MOVW    R6,CLASS_BOX_DSCR       ; Save as the number of bytes in
                     1F6B  6053                                          ;  CLASS_BOX_BUF.
57 0000159B'EF DE    1F6B  6054          MOVAL   CLASS_BUF,R7            ; Get the address of the class buffer.
55 00001193'EF DE    1F72  6055          MOVAL   TITLE_BUF,R5            ; Get the address of the title buffer.
      7C 8F 87 91    1F79  6056 10$:     CMPB    (R7)+,#^A/!/            ; Is this character a vertical bar?
            05 12    1F7D  6057          BNEQU   20$                     ; Branch if not.
         88 2B 90    1F7F  6058          MOVB    #^A/+/,(R8)+            ; Store a plus sign in the buffer.
            03 11    1F82  6059          BRB     30$
         88 2D 90    1F84  6060 20$:     MOVB    #^A/-/,(R8)+            ; Store a minus sign in the buffer.
      7C 8F 85 91    1F87  6061 30$:     CMPB    (R5)+,#^A/!/            ; Is this character a vertical bar?
            05 12    1F8B  6062          BNEQU   40$                     ; Branch if not.
```

```
                        1F8D  6063          MOVB    #^A/+/,(R9)+            ; Store a plus sign in the buffer.
                        1F90  6064          BRB     50$
                        1F92  6065 40$:     MOVB    #^A/-/,(R9)+            ; Store a minus sign in the buffer.
                        1F95  6066 50$:     SOBGTR  R6,10$                  ; Loop for all characters.
                        1F98  6067          POPR    #^M<R5,R6,R7,R8,R9>
                        1F9C  6068          RSB
                        1F9D  6069
                        1F9D  6070
```

```
                     1F9D  6072 ;++
                     1F9D  6073 ;
                     1F9D  6074 ; FUNCTIONAL DESCRIPTION:
                     1F9D  6075 ;
                     1F9D  6076 ;       This routine builds the descriptors for the lines which make up the
                     1F9D  6077 ;       boxes surrounding the classes and titles in the display header and the
                     1F9D  6078 ;       line following the last line of data in the report when the report is
                     1F9D  6079 ;       run on a VT100 compatable terminal.
                     1F9D  6080 ;
                     1F9D  6081 ; CALLING SEQUENCE:
                     1F9D  6082 ;
                     1F9D  6083 ;       JSB     BUILD_DECCRT_BOX
                     1F9D  6084 ;
                     1F9D  6085 ; CALLED BY:
                     1F9D  6086 ;
                     1F9D  6087 ;       GET_HEADER
                     1F9D  6088 ;
                     1F9D  6089 ; OUTPUTS:
                     1F9D  6090 ;
                     1F9D  6091 ;       Date for line above classes stored in CLASS_BOX_BUF.
                     1F9D  6092 ;       Data for line above fields stored in FIELD_TOP_BOX_BUF.
                     1F9D  6093 ;       Data for line below fields stored in FIELD_BOT_BOX_BUF.
                     1F9D  6094 ;       Data for line below the last line of data stored in
                     1F9D  6095 ;       LAST_LINE_BUF for SCS display or LAST_CLUB_LINE_BUF for
                     1F9D  6096 ;       CLUB display.
                     1F9D  6097 ;
                     1F9D  6098 ; RETURN STATUS:
                     1F9D  6099 ;
                     1F9D  6100 ;       None.
                     1F9D  6101 ;
                     1F9D  6102 ; REGISTER USAGE:
                     1F9D  6103 ;
                     1F9D  6104 ;       R2 -    Pointer into CLASS_BOX_BUF.
                     1F9D  6105 ;       R3 -    Pointer into FIELD_TOP_BOX_BUF.
                     1F9D  6106 ;       R4 -    Pointer into FIELD_BOT_BOX_BUF.
                     1F9D  6107 ;       R5 -    Pointer into LAST_LINE_BUF or LAST_CLUB_LINE_BOX.
                     1F9D  6108 ;       R6 -    Counter for the number of characters in the buffers.
                     1F9D  6109 ;       R7 -    Pointer into CLASS_BUF.
                     1F9D  6110 ;       R8 -    Pointer into TITLE_BUF.
                     1F9D  6111 ;--
                     1F9D  6112
                     1F9D  6113 BUILD_DECCRT_BOX:
          01FC 8F    BB   1F9D  6114         PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>
56   0000118B'EF    3C   1FA1  6115         MOVZWL  TITLE_DSCR,R6           ; Get the number of bytes in TITLE_BUF.
                     1FA8  6116         JEQL    60$                     ; Branch if nothing to display.
52   0000076B'EF    DE   1FAD  6117         MOVAL   CLASS_BOX_BUF,R2        ; Get address of buffer for line above
                     1FB4  6118                                         ;  classes.
53   00000973'EF    DE   1FB4  6119         MOVAL   FIELD_TOP_BOX_BUF,R3    ; Get address of buffer for line above
                     1FBB  6120                                         ;  fields.
54   00000B7B'EF    DE   1FBB  6121         MOVAL   FIELD_BOT_BOX_BUF,R4    ; Get address of buffer for line below
                     1FC2  6122                                         ;  fields.
               00    E1   1FC2  6123         BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
  09 00000001'EF         1FC4  6124                 CLASS_SELECT,5$         ;  class.
55   00000F8B'EF    DE   1FCA  6125         MOVAL   LAST_CLUB_LINE_BUF,R5   ; Get address of buffer for line below
                     1FD1  6126                                         ;  the last data line of CLUB display.
               07    11   1FD1  6127         BRB     6$                      ; Branch to common code.
55   00000D83'EF    DE   1FD3  6128 5$:     MOVAL   LAST_LINE_BUF,R5        ; Get address of buffer for line below
```

```
                                1FDA  6129
                                1FDA  6130 6$:      MOVAL   CLASS_BUF,R7            ; Get address of buffer for class
                                1FE1  6131                                          ;  names.
                                1FE1  6132          MOVAL   TITLE_BUF,R8            ; Get address of buffer for field
                                1FE8  6133                                          ;  names.
                                1FE8  6134          MOVZWL  TITLE_DSCR,R6           ; Get the number of bytes in TITLE_BUF.
                             90 1FEF  6135          MOVB    #GRAPHICS_ON,(R2)+      ; Store character to turn on graphics in
                             90 1FF2  6136          MOVB    #GRAPHICS_ON,(R3)+      ;  each of the buffers.
                             90 1FF5  6137          MOVB    #GRAPHICS_ON,(R4)+      ;
                             90 1FF8  6138          MOVB    #GRAPHICS_ON,(R5)+      ;
                             E0 1FFB  6139          BBS     #SCR$V_MORE_LFT,-       ; Branch if screen has been scrolled
                                1FFD  6140                  SCROLL_FLAGS,10$        ;  left.
                             90 2003  6141          MOVB    #UL_CORNER,(R2)+        ; Store upper-left corner.
                             90 2007  6142          MOVB    #LEFT_T,(R3)+           ; Store left "T".
                             90 200B  6143          MOVB    #LEFT_T,(R4)+           ; Store left "T".
                             90 200F  6144          MOVB    #LL_CORNER,(R5)+        ; Store lower-left corner.
                             C0 2013  6145          ADDL    #DELM_DSCR_LEN,R7       ; Ignore first character in CLASS_BUF
                             C0 2016  6146          ADDL    #DELM_DSCR_LEN,R8       ;  and FIELD_BUF. (assumed a ":").
                             C2 2019  6147          SUBL    #DELM_DSCR_LEN,R6       ; Decrement number of bytes remaining.
                             91 201C  6148 10$:     CMPB    #GRAPHICS_ON,(R8)+      ; Is there a field delimiter in
                                201F  6149                                          ;  FIELD_BUF?
                             12 201F  6150          BNEQU   30$                     ; Branch if not.
                             C0 2021  6151          ADDL    #DELM_DSCR_LEN-1,R8     ; Update pointer into FIELD_BUF.
                             C2 2024  6152          SUBL    #DELM_DSCR_LEN-1,R6     ; Update counter.
                             90 2027  6153          MOVB    #CROSS,(R4)+            ; Store crossing lines in
                                202B  6154                                          ;  FIELD_BOT_BOX_BUF.
                             90 202B  6155          MOVB    #BOTTOM_T,(R5)+         ; Store a bottom "T" in LAST_LINE_BUF.
                             91 202F  6156          CMPB    #GRAPHICS_ON,(R7)+      ; Is there an escape sequence in
                                2032  6157                                          ;  CLASS_BUF?
                             12 2032  6158          BNEQU   20$                     ; Branch if not.
                             90 2034  6159          MOVB    #CROSS,(R3)+            ; Store crossing lines in
                                2038  6160                                          ;  FIELD_TOP_BOX_BUF.
                             90 2038  6161          MOVB    #TOP_T,(R2)+            ; Store a top "T" in CLASS_BOX_BUF.
                             C0 203C  6162          ADDL    #DELM_DSCR_LEN-1,R7     ; Update pointer in CLASS_BUF.
                             11 203F  6163          BRB     40$                     ; Join common code.
                                2041  6164
                             90 2041  6165 20$:     MOVB    #TOP_T,(R3)+            ; Store a top "T" in FIELD_TOP_BOX_BUF.
                             90 2045  6166          MOVB    #SCAN_5,(R2)+           ; Store a horizontal line = scan 5 in
                                2049  6167                                          ;  CLASS_BOX_BUF.
                             11 2049  6168          BRB     40$                     ; Join common code.
                                204B  6169
                             90 204B  6170 30$:     MOVB    #SCAN_5,(R2)+           ; Store a horizontal line - scan 5 in
                                204F  6171                                          ;  CLASS_BOX_BUF...
                             90 204F  6172          MOVB    #SCAN_5,(R3)+           ;    in FIELD_TOP_BOX_BUF...
                             90 2053  6173          MOVB    #SCAN_5,(R4)+           ;    in FIELD_BOT_BOX_BUF...
                             90 2057  6174          MOVB    #SCAN_5,(R5)+           ;    and in LAST_LINE_BUF.
                             D6 205B  6175          INCL    R7                      ; Update pointer into CLASS_BUF
                             F5 205D  6176 40$:     SOBGTR  R6,10$                  ; Loop for all bytes in CLASS_BUF.
                             D7 2060  6177          DECL    R2                      ; Now that we know we've hit the end
                             D7 2062  6178          DECL    R3                      ;  backup one character in each buffer
                             D7 2064  6179          DECL    R4                      ;  and fix the last character on the
                             D7 2066  6180          DECL    R5                      ;  line.
                          8F B0 2068  6181          MOVW    #<GRAPHICS_OFF@8>!UR_CORNER,-
                             82       206C  6182                  (R2)+             ; Store an upper-right corner followed
                                206D  6183                                          ;  by character to turn graphics off in
                                206D  6184                                          ;  CLASS_BOX_BUF.
                          8F B0 2069  6185          MOVW    #<GRAPHICS_OFF@8>!RIGHT_T,-
```

```
                      83        2071  6186                    (R3)+                   ; Store a right "T" followed by
                                2072  6187                                            ;  character to turn graphics off
                                2072  6188                                            ;  in FIELD_TOP_BOX_BUF.
                   0F75 8F      B0  2072  6189          MOVW    #<GRAPHICS_OFF@8>!RIGHT_T,-
                        84          2076  6190                  (R4)+                   ; Store a right "T" followed by
                                    2077  6191                                        ;  character to turn graphics off in
                                    2077  6192                                        ;  FIELD_BOT_BOX_BUF.
                   0F6A 8F      B0  2077  6193          MOVW    #<GRAPHICS_OFF@8>!LR_CORNER,-
                        85          207B  6194                  (R5)+                   ; Store a lower-right corner followed
                                    207C  6195                                        ;  by character to turn graphics off in
                                    207C  6196                                        ;  LAST_LINE_BUF.
                                    207C  6197
00000763'EF  52 00000767'EF  C3  207C  6198          SUBL3   CLASS_BOX_DSCR+4,R2,-   ; Save the number of characters stored
                                    2088  6199                  CLASS_BOX_DSCR      ;  in CLASS_BOX_BUF.
                00000960'EF  C3  2088  620C          SUBL3   FIELD_TOP_BOX_DSCR+4,-  ; Save the number of characters stored
          00000960'EF    53        208E  6201                  R3,FIELD_TOP_BOX_DSCR ;  in FIELD_TOP_BOX_BUF.
                00000B77'EF  C3  2094  6202          SUBL3   FIELD_BOT_BOX_DSCR+4,-  ; Save the number of characters stored
          00000B73'EF    54        209A  6203                  R4,FIELD_BOT_BOX_DSCR ;  in FIELD_BOT_BOX_BUF.
                         00  E1  20A0  6204          BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
            0E 00000001'EF        20A2  6205                  CLASS_SELECT,50$      ;  class.
                00000F87'EF  C3  20A8  6206          SUBL3   LAST_CLUB_LINE_DSCR+4,- ; Save the number of characters stored
          00000F83'EF    55        20AE  6207                  R5,LAST_CLUB_LINE_DSCR ;  in LAST_CLUB_LINE_BUF.
                         36  11  20B4  6208          BRB     70$                     ; Branch to common code.
00000D7B'EF  55 00000D7F'EF  C3  20B6  6209 50$:     SUBL3   LAST_LINE_DSCR+4,R5,-   ; Save the number of characters stored
                                    20C2  6210                  LAST_LINE_DSCR      ;  in LAST_LINE_BUF.
                         28  11  20C2  6211          BRB     70$                     ; Branch to common code.
                                    20C4  6212
                00000763'EF  D4  20C4  6213 60$:     CLRL    CLASS_BOX_DSCR          ; Nothing to display, so clear length in
                00000960'EF  D4  20CA  6214          CLRL    FIELD_TOP_BOX_DSCR      ;  all descriptors.
                00000B73'EF  D4  20D0  6215          CLRL    FIELD_BOT_BOX_DSCR      ;
                         00  E1  20D6  6216          BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
            08 00000001'EF        20D8  6217                  CLASS_SELECT,65$      ;  class.
                00000F83'EF  D4  20DE  6218          CLRL    LAST_CLUB_LINE_DSCR     ;
                         06  11  20E4  6219          BRB     70$                     ; Branch to common code.
                00000D7B'EF  D4  20E6  6220 65$:     CLRL    LAST_LINE_DSCR          ;
                     01FC 8F  BA  20EC  6221 70$:     POPR    #^M<R2,R3,R4,R5,R6,R7,R8>
                              05  20FC  6222          RSB
                                  20F1  6223
                                  20F1  6224
```

```
                                20F1  6226          .SBTTL          -       DRAW_LAST_LINE
                                20F1  6227
                                20F1  6228 ;++
                                20F1  6229 ;
                                20F1  6230 ; FUNCTIONAL DESCRIPTION:
                                20F1  6231 ;
                                20F1  6232 ;       This routine is used to output the line immediately following the
                                20F1  6233 ;       data, i.e., the bottom line of the display box.
                                20F1  6234 ;
                                20F1  6235 ; CALLING SEQUENCE:
                                20F1  6236 ;
                                20F1  6237 ;       JSB     DRAW_LAST_LINE
                                20F1  6238 ;
                                20F1  6239 ; CALLED BY:
                                20F1  6240 ;
                                20F1  6241 ;       DISPLAY_CLUB
                                20F1  6242 ;       DISPLAY_COMMON
                                20F1  6243 ;       GET_HELP
                                20F1  6244 ;
                                20F1  6245 ; INPUTS:
                                20F1  6246 ;
                                20F1  6247 ;       LAST_LINE_DSCR or LAST_CLUB_LINE_DSCR filled in for DECcrts.
                                20F1  6248 ;       FIELD_TOP_BOX_DSCR filled in if not DECcrt.
                                20F1  6249 ;
                                20F1  6250 ; OUTPUTS:
                                20F1  6251 ;
                                20F1  6252 ;       One line output to SCREEN_BUFFER.
                                20F1  6253 ;
                                20F1  6254 ; RETURN STATUS:
                                20F1  6255 ;
                                20F1  6256 ;       R0 = Status returned form DRAW_LINE routine.
                                20F1  6257 ;
                                20F1  6258 ; REGISTER USAGE:
                                20F1  6259 ;
                                20F1  6260 ;       R11 points to descriptor for the line to be output.
                                20F1  6261 ;
                                20F1  6262 ;--
                                20F1  6263
                                20F1  6264 DRAW_LAST_LINE:
                     0800 8F    BB  20F1  6265          PUSHR   #^M<R11>
38 00002247'EF       01    E1    20F5  6266          BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if output device isn't a
                                20FD  6267                                          ;  DECcrt terminal.
                     00    E1    20FD  6268          BBC     #RPT$V_CLSTR,-          ; Branch if not displaying the Cluster
     27 00000000'EF               20FF  6269                  CURRENT_REPORT,20$      ;  report.
                     00    E1    2105  6270          BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
     14 00000001'EF               2107  6271                  CLASS_SELECT,10$        ;  class.
   5B   00000F83'EF    DE    210D  6272          MOVAL   LAST_CLUB_LINE_DSCR,R11 ; Point to descriptor for last line
                                2114  6273                                          ;  of CLUB display.
        000021C8'EF    D0    2114  6274          MOVL    CLUB_HEADER_SIZE,-      ; Get size of CLUB header.
        000021C4'EF          211A  6275                  HEADER_SIZE             ;
                     1B    11    211F  6276          BRB     40$                     ; Branch to common code.
        000021CC'EF    D0    2121  6277 10$:     MOVL    SCS_HEADER_SIZE,-       ; Get size of SCS header.
        000021C4'EF          2127  6278                  HEADER_SIZE             ;
   5B   00000D7B'EF    DE    212C  6279 20$:     MOVAL   LAST_LINE_DSCR,R11      ; Point to descriptor for last line
                                2133  6280                                          ;  of SCS display.
                     07    11    2133  6281          BRB     40$                     ; Jump to common code.
   5B   0000096B'EF    DE    2135  6282 30$:     MOVAL   FIELD_TOP_BOX_DSCR,R11  ; Point to descriptor for line below
```

```
                    213C  6283
                                                        ;  the field names.
     0327    30     213C  6284 40$:    BSBW    DRAW_LINE
                                                        ; Output the last line.
0800 8F      BA     213F  6285          POPR    #^M<R11>
             05     2143  6286          RSB
                    2144  6287
                    2144  6288
```

```
                        2144  6290          .SBTTL          -       Field title generation.
                        2144  6291 ;++
                        2144  6292 ;
                        2144  6293 ; FUNCTIONAL DESCRIPTION:
                        2144  6294 ;
                        2144  6295 ;       This routine scans the field descriptor bit map to determine which
                        2144  6296 ;       fields will be displayed.  If a field is to be displayed, then its
                        2144  6297 ;       title is added to the title buffer along with the appropriate
                        2144  6298 ;       delimiters and spaces so that the title is centered in the field.
                        2144  6299 ;       The scan continues until the entire bit map is checked, or the title
                        2144  6300 ;       buffer is full.
                        2144  6301 ;
                        2144  6302 ; CALLING SEQUENCE:
                        2144  6303 ;
                        2144  6304 ;       JSB     GET_TITLES
                        2144  6305 ;
                        2144  6306 ; CALLED BY:
                        2144  6307 ;
                        2144  6308 ;       GET_HEADER
                        2144  6309 ;
                        2144  6310 ; OUTPUTS:
                        2144  6311 ;
                        2144  6312 ;       Field titles are output to TITLE_BUF.  Descriptor for the title is
                        2144  6313 ;       built at TITLE_DSCR.
                        2144  6314 ;
                        2144  6315 ; RETURN STATUS:
                        2144  6316 ;
                        2144  6317 ;       R0 = SS$_NORNAL
                        2144  6318 ;
                        2144  6319 ; REGISTER USAGE:
                        2144  6320 ;
                        2144  6321 ;       R2 -    Used to determine when the located title is in a different
                        2144  6322 ;               class than the previously located title.
                        2144  6323 ;       R3 -    Points to the next free byte in the title buffer.
                        2144  6324 ;       R6 -    Number of character positions left on the line before the
                        2144  6325 ;               screen line overflows.
                        2144  6326 ;       R7 -    Used to hold the size of the current display field.
                        2144  6327 ;       R9 -    Used to calculate the number of spaces needed to center the
                        2144  6328 ;               title in the field.
                        2144  6329 ;       R10 -   Points to the start of the current field descriptor entry.
                        2144  6330 ;               (From FIND_ENTRY routine)
                        2144  6331 ;
                        2144  6332 ;--
                        2144  6333
                        2144  6334 GET_TITLES:
            06CC 8F     BB  2144  6335          PUSHR   #^M<R2,R3,R6,R7,R9,R10>
    00000425'EF         94  2148  6336          CLRB    NUM_FIELDS                      ; Initialize number of fields found.
                00      E1  214E  6337          BBC     #RPT$V_CLSTR,-                  ; Branch if we're not displaying the
 0D 00000000'EF             2150  6338                  CURRENT_REPORT,3$               ;  cluster report.
    00000464'EF         D0  2156  6339          MOVL    CLSTR_H_SCROLL,-                ; Get a copy of the scroll value for the
    0000046C'EF             215C  6340                  H_SCROLL_LINE                   ;  cluster report.
                08      11  2161  6341          BRB     4$                              ; Branch to common code.
    00000468'EF         D0  2163  6342 3$:       MOVL    LOCAL_H_SCROLL,-                ; Get a copy of the scroll value for the
    0000046C'EF             2169  6343                  H_SCROLL_LINE                   ;  local report.
53  000021D0'EF         DE  216E  6344 4$:       MOVAL   TITLE_LEN_ARRAY,R3              ; Set length of titles to 0.
                34      BB  2175  6345          PUSHR   #^M<R2,R4,R5>                   ;
  00    63    00      2C  2177  6346          MOVC5   #0,(R3),#0,-                    ;
```

```
                63   24        217B  6347                    #4*NUM_CLASSES,(R3)     ;
                     34   BA   217D  6348          POPR      #^M<R2,R4,R5>           ;
     0000118B'EF     D4   217F  6349          CLRL      TITLE_DSCR              ; Set length of buffer to 0.
     000021C4'EF     D4   2185  6350          CLRL      HEADER_SIZE             ; Set width of header to 0.
53   00001193'EF     DE   218B  6351          MOVAL     TITLE_BUF,R3            ; Get pointer to title buffer.
000021C0'EF   53     D0   2192  6352          MOVL      R3,TITLE_START          ; Save address of start of titles.
56   00002222'EF     3C   2199  6353          MOVZWL    SCREEN_WIDTH,R6         ; Get screen width.
                     01   E0   21A0  6354          BBS       #SCR$V_MORE_LFT,-       ; Branch if screen has been scrolled
  09 0000047A'EF          21A2  6355                    SCROLL_FLAGS,5$         ;  left.
               0274   30   21A8  6356          BSBW      DELM                    ; Start line with field delimiter.
     000021C4'EF     D6   21AB  6357          INCL      HEADER_SIZE             ; Header size = 1.
00C021BC'EF   56     D0   21B1  6358 5$:      MOVL      R6,TITLE_COUNT          ; Save the number of bytes remaining on
                          21B8  6359                                            ;  the line.
     00002270'EF     D4   21B8  6360          CLRL      START_BIT               ; Start scanning tne field descriptor
                          21BE  6361                                            ;  bitmap at bit zero.
                     00   E1   21BE  6362          BBC       #RPT$V_CLSTR,-          ; Branch if we're not displaying the
  09 00000000'EF          21C0  6363                    CURRENT_REPORT,7$       ;  cluster report.
0000226C'EF   02     D0   21C6  6364          MOVL      #CLUSTER_MAP_LEN,MAP_LEN; Save length of cluster bitmap.
                     07   11   21CD  6365          BRB       10$                     ; Jump to common code.
0000226C'EF   0C     D0   21CF  6366 7$:      MOVL      #LOCAL_MAP_LEN,MAP_LEN  ; Save length of local bitmap.
               018A   30   21D6  6367 10$:     BSBW      FIND_ENTRY              ; Locate a field to be displayed.
             6C 50    E9   21D9  6368          BLBC      R0,80$                  ; Branch if field wasn't found.
     00000425'EF     96   21DC  6369          INCB      NUM_FIELDS              ; Increment number of fields found.
           0C AA     CB   21E2  6370 20$:     BICL3     FD$L_CLASS(R10),-       ; Is this field in a different class?
52   000021F4'EF          21E5  6371                    CURRENT_CLASS,R2        ;
                     02   13   21EB  6372          BEQL      30$                     ; Branch if not.
                     63   10   21ED  6373          BSB       SAVE_TITLE_LEN          ; Save length of the class title.
           0C AA     D3   21EF  6374 30$:     BITL      FD$L_CLASS(R10),-       ; Is this class enabled?
     00000001'EF          21F2  6375                    CLASS_SELECT            ;
                     DD   13   21F7  6376          BEQLU     10$                     ; Branch if not.
               0247   30   21F9  6377          BSBW      ADD_ONE_SPACE           ; Preceed title with atleast one space.
                     48   1F   21FC  6378          BLSSU     70$                     ; Branch if it won't fit.
           18 AA     83   21FE  6379          SUBB3     FD$B_TITLE_SIZE(R10),-  ; Get the difference between the number
             59   6A          2201  6380                    FD$B_FIELD_SIZE(R10),R9 ; of characters in the title and the
                          2203  6381                                            ;  size of the field.
                     02   18   2203  6382          BGEQ      40$                     ; Branch if field size is larger than
                          2205  6383                                            ;  the title.
                     59   94   2205  6384          CLRB      R9                      ; Number of spaces needed before and
                          2207  6385                                            ;  after the title equals zero.
                59   59   9A   2207  6386 40$:     MOVZBL    R9,R9                   ; Make R9 a longword.
59   59       FF 8F   9C   220A  6387          ROTL      #-1,R9,R9               ; Calculate the number of spaces to
                          220F  6388                                            ;  center the title.
               0237   30   220F  6389          BSBW      ADD_SPACES              ; Add the spaces to the buffer.
                     32   1F   2212  6390          BLSSU     70$                     ; Branch if the spaces won't fit on the
                          2214  6391                                            ;  screen.
                03 59   1F   E1   2214  6392          BBC       #31,R9,50$              ; Branch if even number of spaces.
               0228   30   2218  6393          BSBW      ADD_ONE_SPACE           ; Otherwise, add one more.
                     29   1F   221B  6394 50$:     BLSSU     70$                     ; Branch if the won't fit.
                57   18 AA   9A   221D  6395          MOVZBL    FD$B_TITLE_SIZE(R10),R7 ; Assume the full title will fit in the
                          2221  6396                                            ;  field.
             6A   57          91   2221  6397          CMPB      R7,FD$B_FIELD_SIZE(R10) ; Will it really fit?
                     03   15   2224  6398          BLEQ      60$                     ; Branch if it will
                57   6A   9A   2226  6399          MOVZBL    FD$B_FIELD_SIZE(R10),R7 ; Get the size of the field.
                     34   BB   2229  6400 60$:     PUSHR     #^M<R2,R4,R5>
63   19 AA           57   28   222B  6401          MOVC3     R7,FD$T_TITLE(R10),(R3) ; Copy title to title buffer.
                     34   BA   2230  6402          POPR      #^M<R2,R4,R5>
                56   57   82   2232  6403          SUBB      R7,R6                   ; Calculate room left on screen.
```

```
                0F   1F  2235  6404          BLSSU   70$                     ; Branch if it didn't fit.
              020F   30  2237  6405          BSBW    ADD_SPACES              ; Add spaces following the title.
                0A   1F  223A  6406          BLSSU   70$                     ; Branch if the spaces won't fit on the
                                                                             ;  screen.
              0204   30  223C  6408          BSBW    ADD_ONE_SPACE           ; Follow title with one space.
                05   1F  223F  6409          BLSSU   70$                     ; Branch if it didn't fit.
              01DB   30  2241  6410          BSBW    DELM                    ; Add ending delimiter to line.
                90   1E  2244  6411          BGEQU   10$                     ; Loop if there's more room on the
                                                                             ;  line.
                56   D4  2246  6413  70$:    CLRL    R6                      ; Number of columns remaining equals
                                                                             ;  zero.
                08   10  2248  6415  80$:    BSB     SAVE_TITLE_LEN          ; Save the length of the titles.
          50    01   D0  224A  6416          MOVL    #SS$_NORMAL,R0          ; Indicate success.
          06CC  8F   BA  224D  6417          POPR    #^M<R2,R3,R6,R7,R9,R10>
                     05  2251  6418          RSB                             ; Return.
```

```
2252  6422 ;++
2252  6423 ;
2252  6424 ; FUNCTIONAL DESCRIPTION:
2252  6425 ;
2252  6426 ;       This routine is used to save the total number of columns used by all the
2252  6427 ;       titles in a particular class that are to be displayed.  This number is
2252  6428 ;       used by the GET_CLASSES routine to center the classes.  This routine
2252  6429 ;       also saves the total number of columns that will be used by the report.
2252  6430 ;       Note that this routine is called whenever a title to be displayed has
2252  6431 ;       been found that is in a different class than the previously found title
2252  6432 ;       or when the last title to be displayed has been found.
2252  6433 ;
2252  6434 ; CALLING SEQUENCE:
2252  6435 ;
2252  6436 ;       JSB     SAVE_TITLE_LEN
2252  6437 ;
2252  6438 ; CALLED BY:
2252  6439 ;
2252  6440 ;       GET_TITLES
2252  6441 ;
2252  6442 ; INPUTS:
2252  6443 ;
2252  6444 ;       R3 points to the address following the last byte stored in the title
2252  6445 ;       buffer.
2252  6446 ;       R6 contains the number of characters that can be added to the title
2252  6447 ;       buffer before the number of characters in the buffer exceeds the width
2252  6448 ;       of the screen.
2252  6449 ;
2252  6450 ; IMPLICIT INPUTS:
2252  6451 ;
2252  6452 ;       TITLE_COUNT = Number of characters that can be added to the title
2252  6453 ;               as saved from a previous call to this rouine.  TITLE_COUNT
2252  6454 ;               is initially set to the screen width by GET_TITLES.
2252  6455 ;       TITLE_START = Address in the title buffer where the last group of
2252  6456 ;               titles begins as saved from a previous call to this routine.
2252  6457 ;               TITLE_START initially points to the start of the title buffer.
2252  6458 ;
2252  6459 ; OUTPUTS:
2252  6460 ;
2252  6461 ;       HEADER_SIZE updated to reflect the width of the report in characters.
2252  6462 ;
2252  6463 ; IMPLICIT OUTPUTS:
2252  6464 ;
2252  6465 ;       TITLE_COUNT updated.
2252  6466 ;       TITLE_START updated.
2252  6467 ;
2252  6468 ; RETURN STATUS:
2252  6469 ;
2252  6470 ;       None.
2252  6471 ;
2252  6472 ; REGISTER USAGE:
2252  6473 ;
2252  6474 ;       R2 -    Used to index into an array containing the lengths of the
2252  6475 ;               titles in each of the classes.
2252  6476 ;       R4 -    Used to point to the base address of the array.
2252  6477 ;       R3,R5 - Used to calculate the number of bytes stored in the title
2252  6478 ;               buffer.
```

```
                                2252  6479 ;
                                2252  6480 ;--
                                2252  6481
                                2252  6482 SAVE_TITLE_LEN:
                          34    BB  2252  6483          PUSHR   #^M<R2,R4,R5>
                       09   00    EA  2254  6484          FFS     #0,#NUM_CLASSES,-        ; Convert class bit to index value.
          52    000021F4'EF         2257  6485                  CURRENT_CLASS,R2         ;
          54    000021D0'EF    DE   225D  6486          MOVAL   TITLE_LEN_ARRAY,R4       ; Get address of title length array.
    6442  000021BC'EF    56    C3  2264  6487          SUBL3   R6,TITLE_COUNT,(R4)[R2]  ; Save length of titles in characters.
       000021C4'EF  6442    C0  226D  6488          ADDL    (R4)[R2],HEADER_SIZE     ; Add to total size of header in
                                2275  6489                                           ;  characters
    55  53    000021C0'EF    C3  2275  6490          SUBL3   TITLE_START,R3,R5        ; Get actual length of titles in bytes.
          0000118B'EF    55    C0  227D  6491          ADDL    R5,TITLE_DSCR            ; Add length to total length of titles.
                    16   50    E9  2284  6492          BLBC    R0,10$                   ; Skip following if FIND_ENTRY failed.
          000021BC'EF    56    D0  2287  6493          MOVL    R6,TITLE_COUNT           ; Save number of bytes remaining.
          000021C0'EF    53    D0  228E  6494          MOVL    R3,TITLE_START           ; Save starting address for next set of
                                2295  6495                                           ;  titles.
                       0C   AA    D0  2295  6496          MOVL    FD$L_CLASS(R10),-        ; Set new class to current class.
                000021F4'EF         2298  6497                  CURRENT_CLASS            ;
                          34    BA  229D  6498 10$:     POPR    #^M<R2,R4,R5>
                                05  229F  6499          RSB
                                22A0  6500
                                22A0  6501
```

```
                        22A0  6503          .SBTTL          -       Class title generation.
                        22A0  6504 ;++
                        22A0  6505 ;
                        22A0  6506 ; FUNCTIONAL DESCRIPTION:
                        22A0  6507 ;
                        22A0  6508 ;       This routine stores the class titles in CLASS_BUF for the GET_HEADER
                        22A0  6509 ;       routine.  The class titles are stored with the appropriate field
                        22A0  6510 ;       delimiters and spacing such that the class titles will be displayed
                        22A0  6511 ;       centered above the field titles.
                        22A0  6512 ;
                        22A0  6513 ;
                        22A0  6514 ; CALLING SEQUENCE:
                        22A0  6515 ;
                        22A0  6516 ;       JSB     GET_CLASSES
                        22A0  6517 ;
                        22A0  6518 ; CALLED BY:
                        22A0  6519 ;
                        22A0  6520 ;       GET_HEADER
                        22A0  6521 ;
                        22A0  6522 ; OUTPUTS:
                        22A0  6523 ;
                        22A0  6524 ;       Clases are stored in CLASS_BUF.
                        22A0  6525 ;
                        22A0  6526 ; RETURN STATUS:
                        22A0  6527 ;
                        22A0  6528 ;       R0 =    SS$_Normal.
                        22A0  6529 ;
                        22A0  6530 ;
                        22A0  6531 ; REGISTER USAGE:
                        22A0  6532 ;
                        22A0  6533 ;       R3 -    Points to next free character position in CLASS_BUF.
                        22A0  6534 ;       R4 -    Used to calculate the number of leading spaces to precede the
                        22A0  6535 ;               class title.
                        22A0  6536 ;       R5 -    Used as a pointer into the class name strings.
                        22A0  6537 ;       R6 -    Used to calculate the number of trailing spaces to follow the
                        22A0  6538 ;               class title.
                        22A0  6539 ;       R7 -    Points to descriptor containing the class name.
                        22A0  6540 ;       R8 -    Used to calculate the number of spaces needed to center the
                        22A0  6541 ;               class name.
                        22A0  6542 ;       R9 -    Used to calculate the number of spaces needed to center the
                        22A0  6543 ;               class name.
                        22A0  6544 ;
                        22A0  6545 ;       R11-    Used to index into the title length array.
                        22A0  6546 ;
                        22A0  6547 ;--
                        22A0  6548
                        22A0  6549 GET_CLASSES:
            0BFC 8F     BB  22A0  6550          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                   5B   D4  22A4  6551          CLRL    R11                             ; Set counter to 0
53     0000159B'EF      DE  22A6  6552          MOVAL   CLASS_BUF,R3                    ; Get address of class buffer.
56     00002222'EF      3C  22AD  6553          MOVZWL  SCREEN_WIDTH,R6                 ; Get screen width.
57     000021D0'EF      DE  22B4  6554          MOVAL   TITLE_LEN_ARRAY,R7              ; Get address of title length array.
55     0000023B'EF      DE  22BB  6555          MOVAL   CLASS_NAMES,R5                  ; Get address of class descriptors.
            54   85     9A  22C2  6556          MOVZBL  (R5)+,R4                        ; Get number of bytes in first string.
                 01     E0  22C5  6557          BBS     #SCR$V_MORE_LFT,-               ; Branch if screen has been scrolled
  03 0000047A'EF          22C7  6558                  SCROLL_FLAGS,2$                 ;  left.
                014F     30  22CD  6559          BSBW    DELM                            ; Add starting delimiter to line.
```

```
                     65   1F   22D0  6560 2$:     BLSSU   20$                     ; Branch if it won't fit on the line.
 52  00000001'EF     D0   22D2  6561           MOVL    CLASS_SELECT,R2          ; Get mask of classes to be displayed.
00   52   01   5B    ED   22D9  6562 3$:     CMPZV   R11,#1,R2,#0             ; Check if class is enabled.
                     13   12   22DE  6563           BNEQ    5$                       ; Branch if so.
        54   FF A5   9A   22E0  6564 4$:     MOVZBL  -1(R5),R4                ; Get number of bytes
           55   54   C0   22E4  6565           ADDL    R4,R5                    ; Point to next class string.
           54   85   9A   22E7  6566           MOVZBL  (R5)+,R4                 ; Get number of bytes in string.
FFE9 5B    01   08   F1   22EA  6567           ACBL    #CLS$V_ERR,#1,R11,3$     ; Branch if there's more classes to
                0044   31   22F0  6568           BRW     20$                      ;  check.
        58   674B   D0   22F3  6569 5$:     MOVL    (R7)[R11],R8             ; Get length of the titles for this
                     22F7  6570                                                ;  class.
              58   D7   22F7  6571           DECL    R8                       ; Subtract one for the delimeter.
              E5   19   22F9  6572           BLSS    4$                       ; Branch if no titles being displayed.
     59   58   54   C3   22FB  6573           SUBL3   R4,R8,R9                 ; Calculate the number of spaces
                     22FF  6574                                                ;  to appear before and after the
                     22FF  6575                                                ;  class name.
              05   14   22FF  6576           BGTR    6$                       ; Branch if more characters in the
                     2301  6577                                                ;  titles than in the class name.
              59   D4   2301  6578           CLRL    R9                       ; Otherwise, no leading or trailing
                     2303  6579                                                ;  spaces around the class name.
           54   58   D0   2303  6580           MOVL    R8,R4                    ; Make length of class same as titles.
 59   59   FF 8F   9C   2306  6581 6$:     ROTL    #-1,R9,R9                ; Calculate the number of leading
                     230B  6582                                                ;  spaces to appear before the class
                     230B  6583                                                ;  type in order to center it.
                013B   30   230B  6584           BSBW    ADD_SPACES               ; Add leading spaces.
                  27   1F   230E  6585           BLSSU   20$                      ; Branch if the spaces won't fit on the
                     2310  6586                                                ;  screen.
        05 59    1F   E5   2310  6587           BBCC    #31,R9,10$               ; Branch if even number of spaces.
                012C   30   2314  6588           BSBW    ADD_ONE_SPACE            ; Add one more space.
                  1E   1F   2317  6589           BLSSU   20$                      ; Branch if it won't fit.
                  34   BB   2319  6590 10$:     PUSHR   #^M<R2,R4,R5>
     63   65   54   28   231B  6591           MOVC3   R4,(R5),(R3)             ; Add class type.
                  34   BA   231F  6592           POPR    #^M<R2,R4,R5>
                0125   30   2321  6593           BSBW    ADD_SPACES               ; Add trailing spaces.
                  11   1F   2324  6594           BLSSU   20$                      ; Branch if the spaces won't fit on the
                     2326  6595                                                ;  screen.
                00F6   30   2326  6596           BSBW    DELM                     ; Add ending delimiter.
                  0C   1F   2329  6597           BLSSU   20$                      ; Branch if it won't fit.
        00002222'EF   56   B1   232B  6598 17$:     CMPW    R6,SCREEN_WIDTH          ; Will classes fit on the screen?
                     2332  6599           JLEQU   4$                       ; Branch if they will.
00001593'EF   53   0000159B'8F   C3   2337  6600 20$:     SUBL3   #CLASS_BUF,R3,CLASS_DSCR ; Save length of buffer.
                0BFC 8F   BA   2343  6601           POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                     05   2347  6602           RSB
                     2348  6603
                     2348  6604
```

```
                     2348  6606          .SBTTL  REFRESH routine.
                     2348  6607 ;++
                     2348  6608 ;
                     2348  6609 ; FUNCTIONAL DESCRIPTION:
                     2348  6610 ;
                     2348  6611 ;       This routine causes the display header, which is not normally redraw on
                     2348  6612 ;       each update, to be redraw.
                     2348  6613 ;
                     2348  6614 ; CALLING SEQUENCE:
                     2348  6615 ;
                     2348  6616 ;       CALLS   #0,REFRESH
                     2348  6617 ;
                     2348  6618 ; CALLED BY:
                     2348  6619 ;
                     2348  6620 ;       READ_LOOP
                     2348  6621 ;       READ_WO_PROMPT
                     2348  6622 ;       SHOW_CLUSTER
                     2348  6623 ;       SHOW_LOCAL
                     2348  6624 ;       ADD_FIELD
                     2348  6625 ;       REM_FIELD
                     2348  6626 ;       INCLUDE_CLASS
                     2348  6627 ;       EXCLUDE_CLASS
                     2348  6628 ;       SET_WIDTH
                     2348  6629 ;       SET_SCREEN
                     2348  6630 ;
                     2348  6631 ; OUTPUTS:
                     2348  6632 ;
                     2348  6633 ;       RIP (Refresh in Progress) flag set at MISC_FLAGS.
                     2348  6634 ;
                     2348  6635 ;--
                     2348  6636
                0004 2348  6637 .ENTRY  REFRESH,^M<R2>
                     234A  6638          JBS     #MF$V_IN_OPEN,-         ; Skip refresh if executing from a
                     234A  6639                  MISC_FLAGS,99$          ;  command procedure.
          000021AB'EF   94  2355  6640          CLRB    SWITCH                  ; Initialize switch count.
    00000476'EF   08    C8  235B  6641          BISL    #MF$M_RIP,MISC_FLAGS    ; Indicate refresh in progress.
                        04  2362  6642 99$:     RET
                            2363  6643
                            2363  6644
```

```
2363  6646          .SBTTL  FIND_ENTRY routine.
2363  6647 ;++
2363  6648 ;
2363  6649 ; FUNCTIONAL DESCRIPTION:
2363  6650 ;
2363  6651 ;       This routine is used to scan one of the class bitmaps for a set bit,
2363  6652 ;       indicating that a field is to be displayed.  It returns the address
2363  6653 ;       of the field descriptor to the calling routine.
2363  6654 ;
2363  6655 ;       This routine also performs the horizontal scrolling by ignoring the
2363  6656 ;       first entries found as determined by the value stored in H_SCROLL_LINE.
2363  6657 ;
2363  6658 ; CALLING SEQUENCE:
2363  6659 ;
2363  6660 ;       JSB     FIND_ENTRY
2363  6661 ;
2363  6662 ; CALLED BY:
2363  6663 ;
2363  6664 ;       GET_TITLES
2363  6665 ;       GET_DATA
2363  6666 ;       ADD_FIELD
2363  6667 ;
2363  6668 ; IMPLICIT INPUTS:
2363  6669 ;
2363  6670 ;       ADRS_TABLE -    Contains the address of the table of entry addresses.
2363  6671 ;       BITMAP_ADRS -   Contains the address of bitmap to be searched.
2363  6672 ;       MAP_LEN -       Contains the length of the bitmap.
2363  6673 ;       START_BIT -     Contains the number of the bit in the bitmap to start
2363  6674 ;                       the search.
2363  6675 ;                        (Should be zero on the first call to this routine.)
2363  6676 ;       H_SCROLL_LINE   Set to the number of entries to skip before success is
2363  6677 ;                       returned.
2363  6678 ;
2363  6679 ; OUTPUTS:
2363  6680 ;
2363  6681 ;       R10 =   Address of field descriptor.
2363  6682 ;
2363  6683 ; IMPLICIT OUTPUTS:
2363  6684 ;
2363  6685 ;       START_BIT updated to point to next bit to be checked.
2363  6686 ;       MAP_LEN   updated to indicate the remaining length of the bitmap.
2363  6687 ;
2363  6688 ; RETURN STATUS:
2363  6689 ;
2363  6690 ;       R0 =    SS$_Normal if an entry is found.
2363  6691 ;
2363  6692 ; REGISTER USAGE:
2363  6693 ;
2363  6694 ;       R6  -   Pointer into the address table for the report.
2363  6695 ;       R7  -   Pointer into the bitmap for the report.
2363  6696 ;       R9  -   Used to determine the starting bit in the longword to scan.
2363  6697 ;       R10 -   Used to determine the number of bits left to scan in the
2363  6698 ;               longword.
2363  6699 ;           -   Receives the address of the field descriptor once a entry has
2363  6700 ;               been found.
2363  6701 ;--
2363  6702
```

```
                          2363  6703 FIND_ENTRY:
                03C0 8F   BB  2363  6704          PUSHR   #^M<R6,R7,R8,R9>
                      50  D4  2367  6705          CLRL    R0                      ; Assume failure.
          56  00002264'EF  D0  2369  6706          MOVL    ADRS_TABLE,R6           ; Get address of address table.
          57  00002268'EF  D0  2370  6707          MOVL    BITMAP_ADRS,R7          ; Get address of bitmap.
          59  00002270'EF  D0  2377  6708 5$:      MOVL    START_BIT,R9            ; Get the starting bit position.
                      5A  D4  237E  6709          CLRL    R10                     ; Dividend must be a quadword.
       5A   51   59   20  7B  2380  6710          EDIV    #32,R9,R1,R10           ; Calculate the number of bits left to
            5A   20   5A  C3  2385  6711          SUBL3   R10,#32,R10             ;  scan in the longword.
       59   67   5A   59  EA  2389  6712          FFS     R9,R10,(R7),R9          ; Scan for a set bit.
                      12  12  238E  6713          BNEQ    20$                     ; Branch if found in this longword.
        0000226C'EF   51  D1  2390  6714          CMPL    R1,MAP_LEN              ; Any more map left to scan?
                      32  13  2397  6715          BEQL    99$                     ; Branch if not.
        00002270'EF   59  D0  2399  6716          MOVL    R9,START_BIT            ; Save starting bit for next longword.
                      D5  11  23A0  6717          BRB     5$                      ; Scan next longword.
              5A   6649   D0  23A2  6718 20$:     MOVL    (R6)[R9],R10            ; Get the address from the table.
 00002270'EF   59   01    C1  23A6  6719          ADDL3   #1,R9,START_BIT         ; Save starting bit for next
                          23AE  6720                                              ;  call to FIND_ENTRY.
                   0C AA  D3  23AE  6721          BITL    FD$L_CLASS(R10),-       ; Is this entry in a selected class?
             00000001'EF      23B1  6722                  CLASS_SELECT            ;
                      BF  13  23B6  6723          BEQLU   5$                      ; Branch if not.
             0000046C'EF  D5  23B8  6724          TSTL    H_SCROLL_LINE           ; Is this field scrolled off the left of
                          23BE  6725                                              ;  the screen?
                      08  19  23BE  6726          BLSS    30$                     ; Branch if not.
             0000046C'EF  D7  23C0  6727          DECL    H_SCROLL_LINE           ; Count down the number of fields to be
                          23C6  6728                                              ;  ignored.
                      AF  18  23C6  6729          BGEQ    5$                      ; Ignore this field.
                50   01  D0  23C8  6730 30$:     MOVL    #SS$_NORMAL,R0          ; Indicate success.
                03C0 8F   BA  23CB  6731 99$:     POPR    #^M<R6,R7,R8,R9>        ;
                      05  23CF  6732          RSB                             ;
                          23D0  6733
                          23D0  6734
```

```
                        23D0  6736          .SBTTL  Miscellaneous routines.
                        23D0  6737 ;++
                        23D0  6738 ;
                        23D0  6739 ; FUNCTIONAL DESCRIPTION:
                        23D0  6740 ;
                        23D0  6741 ;       Routine to clear the terminal's screen.
                        23D0  6742 ;
                        23D0  6743 ; CALLING SEQUENCE:
                        23D0  6744 ;
                        23D0  6745 ;       JSB     CLEAR_SCREEN
                        23D0  6746 ;
                        23D0  6747 ; CALLED BY:
                        23D0  6748 ;
                        23D0  6749 ;       MAIN_ERR
                        23D0  6750 ;       SHOW_CLUSTER
                        23D0  6751 ;       SHOW_LOCAL
                        23D0  6752 ;
                        23D0  6753 ; IMPLICIT OUTPUTS:
                        23D0  6754 ;
                        23D0  6755 ;       COL_NUM (column number) and LINE_NUM (line number) are set to 1
                        23D0  6756 ;
                        23D0  6757 ; RETURN STATUS:
                        23D0  6758 ;
                        23D0  6759 ;       R0 = SS$_NORMAL or failing status from SRCPKG routine.
                        23D0  6760 ;
                        23D0  6761 ;
                        23D0  6762 ;--
                        23D0  6763
                        23D0  6764 CLEAR_SCREEN:
            50    01    D0  23D0  6765          MOVL    #SS$_NORMAL,R0          ; Assume success.
   0000224C'EF    01    D0  23D3  6766          MOVL    #1,COL_NUM              ; Start at column 1
   00002248'EF    01    D0  23DA  6767          MOVL    #1,LINE_NUM             ;  and line 1.
13 00002247'EF    02    E0  23E1  6768          BBS     #DVT$V_HARD,OUT_DEV,99$ ; Skip if output is hardcopy device.
        0000224C'EF    3F  23E9  6769          PUSHAW  COL_NUM                 ; Clear the screen.
        00002248'EF    3F  23EF  6770          PUSHAW  LINE_NUM                ;
   00000000'GF    02    FB  23F5  6771          CALLS   #2,G^LIB$ERASE_PAGE     ;
                        05  23FC  6772 99$:     RSB                             ; Return with status.
                            23FD  6773
                            23FD  6774
```

```
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Routines to erase one or two lines on the terminal.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB     ERASE_LINE
                        ;       JSB     ERASE_TWO_LINES
                        ;
                        ; CALLED BY:
                        ;
                        ;       ERASE_PAGE
                        ;       DRAW_LINE
                        ;
                        ;       READ_WO_PROMPT
                        ;       DISPLAY_PTP
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       LINE_NUM (line number) is incremented.
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       R0 = SS$_NORMAL or failing status from SRCPKG routine.
                        ;
                        ;
                        ;--
                                .ENABL  LSB

                        ERASE_TWO_LINES:
                03  10          BSB     ERASE_LINE              ; Erase a line.
              1C 50  E9          BLBC    R0,99$                  ; Branch on error.

                        ERASE_LINE:
              50  01  D0        MOVL    #SS$_NORMAL,R0          ; Assume success.
  0000224C'EF  3F              PUSHAW  COL_NUM                 ; Erase the line
  00002248'EF  3F              PUSHAW  LINE_NUM                ;
00000000'GF  02  FB            CALLS   #2,G^LIB$ERASE_LINE     ;
  00002248'EF  D6              INCL    LINE_NUM                ; Bump line number.
               05              RSB                             ; Return with status.

                                .DSABL  LSB
```

```
                        241F  6822 ;++
                        241F  6823 ;
                        241F  6824 ; FUNCTIONAL DESCRIPTION:
                        241F  6825 ;
                        241F  6826 ;       The following routine adds a control string to a line of data to
                        241F  6827 ;        indicate the start, and end of fields in the data line.
                        241F  6828 ;
                        241F  6829 ; CALLING SEQUENCE:
                        241F  6830 ;
                        241F  6831 ;       JSB     DELM
                        241F  6832 ;
                        241F  6833 ; CALLED BY:
                        241F  6834 ;
                        241F  6835 ;       GET_TITLES
                        241F  6836 ;       GET_CLASSES
                        241F  6837 ;
                        241F  6838 ; INPUTS:
                        241F  6839 ;
                        241F  6840 ;       R3 -    Address of the next free character position in the buffer
                        241F  6841 ;       R6 -    Number of bytes that can be added to the buffer before its
                        241F  6842 ;               length exceeds that of the screen line.
                        241F  6843 ;
                        241F  6844 ; OUTPUTS:
                        241F  6845 ;
                        241F  6846 ;       R3 -    Next free position in the buffer.
                        241F  6847 ;       R6 -    Updated number of bytes that can be added to the buffer before
                        241F  6848 ;               its length exceeds that of the screen line.
                        241F  6849 ;
                        241F  6850 ;
                        241F  6851 ; RETURN STATUS:
                        241F  6852 ;
                        241F  6853 ;       R0 =    SS$_NORMAL.
                        241F  6854 ;
                        241F  6855 ;--
                        241F  6856
                        241F  6857 DELM:
                    34  BB  241F  6858          PUSHR   #^M<R2,R4,R5>
        06 00002247'EF  01  E0  2421  6859          BBS     #DVT$V_DCRT,OUT_DEV,10$ ; Branch if terminal is a DECcrt.
                 83 7C  8F  90  2429  6860          MOVB    #^A/¦/,(R3)+            ; Store a VT52 vertical bar "¦".
                    0C  11  242D  6861          BRB     20$                     ; Jump to common code.
0000017B'EF 00000173'EF  28  242F  6862 10$:    MOVC3   DELM_DSCR,DELM_DSCR+8,- ; Store a DECcrt vertical bar.
                    63      243A  6863          (R3)                    ;
                    56  D7  243B  6864 20$:    DECL    R6                      ; Subtract the length of the string
                        243D  6865                                          ;  in characters as it will appear on
                        243D  6866                                          ;  the screen.
                    34  BA  243D  6867          POPR    #^M<R2,R4,R5>
                 50 01  D0  243F  6868          MOVL    #SS$_NORMAL,R0
                    05      2442  6869          RSB
                        2443  6870
                        2443  6871
```

```
                     2443  6873 ;++
                     2443  6874 ;
                     2443  6875 ; FUNCTIONAL DESCRIPTION:
                     2443  6876 ;
                     2443  6877 ;       This routine adds a single space to a buffer at the location pointed to
                     2443  6878 ;       by R3.
                     2443  6879 ;
                     2443  6880 ; CALLING SEQUENCE:
                     2443  6881 ;
                     2443  6882 ;       JSB     ADD_ONE_SPACE
                     2443  6883 ;
                     2443  6884 ; CALLED BY:
                     2443  6885 ;
                     2443  6886 ;       GET_TITLES
                     2443  6887 ;       GET_CLASSES
                     2443  6888 ;
                     2443  6889 ; INPUTS:
                     2443  6890 ;
                     2443  6891 ;       R3 -    Points to the next free position in the buffer.
                     2443  6892 ;       R6 -    Contains the number of characters that can be output to the
                     2443  6893 ;               buffer before the number of characters in the buffer exceeds
                     2443  6894 ;               the width of the screen.
                     2443  6895 ;
                     2443  6896 ; OUTPUTS:
                     2443  6897 ;
                     2443  6898 ;       Single space stored in buffer pointed to by R3.
                     2443  6899 ;
                     2443  6900 ; IMPLICIT OUTPUTS:
                     2443  6901 ;
                     2443  6902 ;       R3 and R6 updated
                     2443  6903 ;
                     2443  6904 ; RETURN STATUS:
                     2443  6905 ;
                     2443  6906 ;       None.
                     2443  6907 ;
                     2443  6908 ;--
                     2443  6909
                     2443  6910 ADD_ONE_SPACE:
83   20   90         2443  6911          MOVB    #^A/ /,(R3)+            ; Add one space to the buffer.
     56   D7         2446  6912          DECL    R6                      ; Decrease room left on display line.
          05         2448  6913          RSB
                     2449  6914
```

```
                     2449  6916 ;++
                     2449  6917 ;
                     2449  6918 ; FUNCTIONAL DESCRIPTION:
                     2449  6919 ;
                     2449  6920 ;       This routine adds a number of spaces to a buffer by recusively calling
                     2449  6921 ;       ADD_ONE_SPACE.
                     2449  6922 ;
                     2449  6923 ; CALLING SEQUENCE:
                     2449  6924 ;
                     2449  6925 ;       JSB     ADD_SPACES
                     2449  6926 ;
                     2449  6927 ; CALLED BY:
                     2449  6928 ;
                     2449  6929 ;       GET_TITLES
                     2449  6930 ;       GET_CLASSES
                     2449  6931 ;
                     2449  6932 ; INPUTS:
                     2449  6933 ;
                     2449  6934 ;       R9 -    Number of spaces to output.
                     2449  6935 ;       R3 -    Points to the next free position in the buffer.
                     2449  6936 ;       R6 -    Contains the number of characters that can be output to the
                     2449  6937 ;               buffer before the number of characters in the buffer exceeds
                     2449  6938 ;               the width of the screen.
                     2449  6939 ;
                     2449  6940 ; OUTPUTS:
                     2449  6941 ;
                     2449  6942 ;       Spaces stored in buffer pointed to by R3.
                     2449  6943 ;
                     2449  6944 ; IMPLICIT OUTPUTS:
                     2449  6945 ;
                     2449  6946 ;       R3 and R6 updated
                     2449  6947 ;
                     2449  6948 ; RETURN STATUS:
                     2449  6949 ;
                     2449  6950 ;       None.
                     2449  6951 ;
                     2449  6952 ;--
                     2449  6953
                     2449  6954 ADD_SPACES:
0200 8F    BB        2449  6955          PUSHR   #^M<R9>
     59    95        244D  6956          TSTB    R9                      ; Any spaces to be output?
     05    12        244F  6957          BNEQ    10$                     ; Branch if so.
59   01    D0        2451  6958          MOVL    #1,R9                   ; Otherwise clear zero flag.
     0B    11        2454  6959          BRB     99$                     ; And return.
     EB    10        2456  6960 10$:     BSB     ADD_ONE_SPACE           ; Output one space.
     07    1F        2458  6961          BLSSU   99$                     ; Return if there's no more room on the
                     245A  6962                                          ;  line.
     59    97        245A  6963          DECB    R9                      ; Count down the number of spaces to
                     245C  6964                                          ;  output.
     F8    12        245C  6965          BNEQ    10$                     ; Loop till all spaces are output.
59   01    D0        245E  6966          MOVL    #1,R9                   ; Clear zero flag.
0200 8F    BA        2461  6967 99$:     POPR    #^M<R9>
           05        2465  6968          RSB
                     2466  6969
                     2466  6970
```

```
                        2466  6972 ;++
                        2466  6973 ;
                        2466  6974 ; FUNCTIONAL DESCRIPTION:
                        2466  6975 ;
                        2466  6976 ;       This routine outputs a line to the screen buffer.  If the line causes
                        2466  6977 ;       the screen buffer to overflow then the screen buffer is output to the
                        2466  6978 ;       screen and the line output retried.
                        2466  6979 ;
                        2466  6980 ;       This routine also handles the case where the output line is longer than
                        2466  6981 ;       the width of the screen by outputing a trunction character at the end
                        2466  6982 ;       of the line.  The truncation character is either a block or a diamond
                        2466  6983 ;       shape depending on terminal type.
                        2466  6984 ;
                        2466  6985 ; CALLING SEQUENCE:
                        2466  6986 ;
                        2466  6987 ;       JSB     DRAW_LINE
                        2466  6988 ;
                        2466  6989 ; CALLED BY:
                        2466  6990 ;
                        2466  6991 ;       DRAW_LAST_LINE
                        2466  6992 ;       GET_HEADER
                        2466  6993 ;       GET_DATA
                        2466  6994 ;
                        2466  6995 ; INPUTS:
                        2466  6996 ;
                        2466  6997 ;       Address of output line descriptor in R11.
                        2466  6998 ;
                        2466  6999 ; IMPLICIT OUTPUTS:
                        2466  7000 ;
                        2466  7001 ;       LINE_NUM = LINE_NUM + 1
                        2466  7002 ;
                        2466  7003 ; RETURN STATUS:
                        2466  7004 ;
                        2466  7005 ;       R0 =    SS$_NORMAL.
                        2466  7006 ;
                        2466  7007 ;--
                        2466  7008
                        2466  7009 DRAW_LINE:
                 0000224C'EF    3F  2466  7010          PUSHAW  COL_NUM                 ; Output the line to the screen buffer.
                 00002248'EF    3F  246C  7011          PUSHAW  LINE_NUM                ;
                         5B     DD  2472  7012          PUSHL   R11                     ;
           00000000'GF   03     FB  2474  7013          CALLS   #3,G^LIB$PUT_SCREEN     ;
                      0F 50     E8  247B  7014          BLBS    R0,10$                  ; Branch if no error.
           00000000'GF   00     FB  247E  7015          CALLS   #0,G^LIB$PUT_BUFFER     ; Output the screen buffer to the
                                    2485  7016                                          ;  screen.
                                    2485  7017          JLBC    R0,99$                  ; Branch on error.
                         D9     11  248B  7018          BRB     DRAW_LINE               ; Try outputing the line again.
0000224C'EF    01    000021C4'EF    A1  248D  7019 10$:     ADDW3   HEADER_SIZE,#1,COL_NUM  ; Erase the remainer of the line.
                       FF66     30  2499  7020          BSBW    ERASE_LINE              ;
                                    249C  7021          JLBC    R0,99$                  ; Branch on error.
     00002222'EF    0000224C'EF    B1  24A2  7022          CMPW    COL_NUM,SCREEN_WIDTH    ; Does the line fit on the screen?
                         19     14  24AD  7023          BGTR    20$                     ; Branch if not.
                         01     8A  24AF  7024          BICB    #SCR$M_MORE_RT,-        ; Clear the flag that indicates that
                 0000047A'EF        24B1  7025                  SCROLL_FLAGS            ;  there is more data off the right of
                                    24B6  7026                                          ;  the screen.
                         93     24B6  7027          BITB    #<SCR$M_MORE_LFT!-      ; Are any of the other scroll flags
                                    24B7  7028                  SCR$M_MORE_UP!-         ;  set?
```

```
                        24B7  7029                      SCR$M_MORE_DWN>,-       ;
                        24B7  7030                      SCROLL_FLAGS            ;
                  0E    24B7  7030                      SCROLL_FLAGS            ;
                  10  12  24BD  7031              BNEQ    40$                     ; Branch if not.
0000047A'EF       10  8A  24BF  7032              BICB    #SCR$M_MORE,SCROLL_FLAGS; Otherwise, be sure to clear the
                        24C6  7033                                              ;  scroll summary flag.
                  07  11  24C6  7034              BRB     40$                     ; Jump to common.
                  11  88  24C8  7035  20$:       BISB    #SCR$M_MORE_RT!SCR$M_MORE,-; Set the scroll flag to indicate
          0000047A'EF       24CA  7036                      SCROLL_FLAGS            ;  that there is more data off the
                        24CF  7037                                              ;  right of the screen.
00000476'EF       04  E0  24CF  7038  40$:       BBS     #MF$V_CONT,MISC_FLAGS,- ; Branch if continuous display.
                  41        24D6  7039                      99$
      39 00002247'EF      01  E1  24D7  7040              BBC     #DVT$V_DCRT,OUT_DEV,99$ ; Branch if terminal isn't a DECcrt.
00002258'EF   00002248'EF     D1  24DF  7041              CMPL    LINE_NUM,LAST_DATA_LINE ; Have we hit the bottom?
                  2C  12  24EA  7042              BNEQ    99$                     ; Branch if not.
                F4ED  30  24EC  7043              BSBW    SET_REGION_DATA         ; Set scrolling region to include only
                        24EF  7044                                              ;  the data area of the screen.
          0000224C'EF   3F  24EF  7045              PUSHAW  COL_NUM                 ; Set cursor back to last line.
          00002248'EF   3F  24F5  7046              PUSHAW  LINE_NUM                ;
00000000'GF       02  FB  24FB  7047              CALLS   #2,G^LIB$SET_CURSOR     ;
                13 50  E9  2502  7048              BLBC    R0,99$                  ; Branch on error.
00000000'GF       00  FB  2505  7049              CALLS   #0,G^LIB$UP_SCROLL      ; Scroll up one line.
                09 50  E9  250C  7050              BLBC    R0,99$                  ; Branch on error.
                F4BC  30  250F  7051              BSBW    SET_REGION_FULL         ; Reset the scrolling region to use the
                        2512  7052                                              ;  full screen.
          00002248'EF   D7  2512  7053              DECL    LINE_NUM                ; Adjust line number.
0000224C'EF       01  D0  2518  7054  99$:       MOVL    #1,COL_NUM              ; Reset column counter to first column.
                  05  251F  7055              RSB
                        2520  7056
                        2520  7057
```

```
                     2520  7059 ;++
                     2520  7060 ;
                     2520  7061 ; FUNCTIONAL DESCRIPTION:
                     2520  7062 ;
                     2520  7063 ;       This routine is used to get a line of data with all the fields filled
                     2520  7064 ;       with spaces.  This is done by disabling the display of all classes and
                     2520  7065 ;       calling the GET_DATA routine.
                     2520  7066 ;
                     2520  7067 ; CALLING SEQUENCE:
                     2520  7068 ;
                     2520  7069 ;       JSB     GET_BLANK
                     2520  7070 ;
                     2520  7071 ; CALLED BY:
                     2520  7072 ;
                     2520  7073 ;       CLUSTER_REPORT
                     2520  7074 ;
                     2520  7075 ; OUTPUTS:
                     2520  7076 ;
                     2520  7077 ;       One line of data output to the screen buffer.
                     2520  7078 ;
                     2520  7079 ; RETURN STATUS:
                     2520  7080 ;
                     2520  7081 ;       None.
                     2520  7082 ;
                     2520  7083 ; REGISTER USAGE:
                     2520  7084 ;
                     2520  7085 ;       R4 -    Used to save a mask of currently enabled classes.
                     2520  7086 ;
                     2520  7087 ;--
                     2520  7088
                     2520  7089 GET_BLANK:
08 0000043C'EF   00002248'EF   E2   2520  7090          BBSS    LINE_NUM,BLANK_LINES,10$; Indicate that this is a blank line.
        14 00000476'EF   03    E0   252C  7091          BBS     #MF$V_RIP,MISC_FLAGS,20$; If refresh in progress, make sure
                                    2534  7092                                          ;  we draw all the lines.
           00002248'EF    E1       2534  7093 10$:     BBC     LINE_NUM,-              ; Branch this line wasn't blank
             08 00000440'EF         253A  7094          PRIOR_BLANK_LINES,20$           ;  during the last pass.
           00002248'EF    D6       2540  7095          INCL    LINE_NUM                ; Otherwise just bump the line number.
                     1D   11       2546  7096          BRB     99$                     ; And finish up.
           000021F8'EF    DD       2548  7097 20$:     PUSHL   DATA_CNTRL              ; Save a copy of the current classes.
           000021F8'EF    D4       254E  7098          CLRL    DATA_CNTRL              ; Disable all classes.
        00002729'EF    00    FB   2554  7099          CALLS   #0,GET_DATA             ; Get blank line of data.
                  07   50    E9   255B  7100          BLBC    R0,99$                  ; Branch on error.
           000021F8'EF 8ED0    255E  7101          POPL    DATA_CNTRL              ; Reenable classes.
                        05     2565  7102 99$:     RSB
                               2566  7103
                               2566  7104
```

```
                     2566  7106 ;++
                     2566  7107 ;
                     2566  7108 ; FUNCTIONAL DESCRIPTION:
                     2566  7109 ;
                     2566  7110 ;       This routine simply calls the cluster display set up routine if
                     2566  7111 ;       the local ports report is being displayed or it calls the local port
                     2566  7112 ;       display set up routine if the cluster report is being displayed.
                     2566  7113 ;
                     2566  7114 ; CALLING SEQUENCE:
                     2566  7115 ;
                     2566  7116 ;       JSB     SWITCH_DISPLAY
                     2566  7117 ;
                     2566  7118 ; CALLED BY:
                     2566  7119 ;
                     2566  7120 ;
                     2566  7121 ;--
                     2566  7122
                     2566  7123 SWITCH_DISPLAY:
                     2566  7124
              00  E0 2566  7125         BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
 09 00000000'EF      2568  7126                 CURRENT_REPORT,10$      ;  cluster report.
00000929'EF   00  FB 256E  7127         CALLS   #0,SHOW_CLUSTER         ; Display the cluster report.
              07  11 2575  7128         BRB     99$                     ; Jump to common code.
0000099F'EF   00  FB 2577  7129 10$:    CALLS   #0,SHOW_LOCAL           ; Display the local report.
              05  257E  7130 99$:    RSB
                  257F  7131
                  257F  7132
```

```
                                K 14
257F  7134          .SBTTL  Data formatting routines.
257F  7135 ;++
257F  7136 ;
257F  7137 ; FUNCTIONAL DESCRIPTION:
257F  7138 ;
257F  7139 ;       This routine formats the display data for the GET_DATA routine.  It
257F  7140 ;       returns the formatted data in FORMAT_BUF and the number of characters
257F  7141 ;       worth of display data in FORMAT_BUF_LEN.
257F  7142 ;
257F  7143 ;       The field descriptor parameter FD$B_FORMAT determines if the routine
257F  7144 ;       will format the output in ascii, hexadecimal, or decimal.  Up to a
257F  7145 ;       quadword of data can be formatted in decimal.  Ascii and hexadecimal
257F  7146 ;       formatting has no restrictions.
257F  7147 ;
257F  7148 ; CALLING SEQUENCE:
257F  7149 ;
257F  7150 ;       JSB     FORMAT_DATA
257F  7151 ;
257F  7152 ; CALLED BY:
257F  7153 ;
257F  7154 ;       DISPLAY_PTP
257F  7155 ;       OUTPUT_DDM
257F  7156 ;       GET_DATA
257F  7157 ;       DISPLAY_FIELD
257F  7158 ;       DISPLAY_STRING
257F  7159 ;       DISPLAY_CIRCUITS
257F  7160 ;       DISPLAY_INCN_TIME
257F  7161 ;       DISPLAY_PROC_NAME
257F  7162 ;       DISPLAY_PORT_STATUS
257F  7163 ;       DISPLAY_PORT_TYPE
257F  7164 ;
257F  7165 ; INPUTS:
257F  7166 ;
257F  7167 ;       R10     = Address of field descriptor.
257F  7168 ;
257F  7169 ; OUTPUTS:
257F  7170 ;
257F  7171 ;       Formated data string returned in FORMAT_BUF.
257F  7172 ;       Length of formatted data string returned in FORMAT_BUF_LEN.
257F  7173 ;
257F  7174 ;
257F  7175 ; RETURN STATUS:
257F  7176 ;
257F  7177 ;       R0 =    SS$_NORMAL or failed status.
257F  7178 ;
257F  7179 ;
257F  7180 ; REGISTER USAGE:
257F  7181 ;
257F  7182 ;       R3 -    Points to the next free position in FORMAT_BUF where the
257F  7183 ;               the conversion is temporarily stored.
257F  7184 ;       R5 -    Contains the byte returned from the CONVERT_DEC routine.
257F  7185 ;       R6 -    Used to index into the data.
257F  7186 ;       R7 -    Points to the address of the data.
257F  7187 ;       R8 -    Used by HEX_DISPLAY to convert a byte to two hex digits.
257F  7188 ;          -    Used by DEC_DISPLAY to hold value from conversion table.
257F  7189 ;       R10 -   Points to the field descriptor.
257F  7190 ;       R11 -   Used to calculate the length of the conversion.
```

```
                        257F  7191 ;       R10,R11 Used by DEC_DISPLAY to hold the quadword of data to be
                        257F  7192 ;               converted to decimal.
                        257F  7193 ;
                        257F  7194 ;--
                        257F  7195
                        257F  7196 FORMAT_DATA:
                0DE8 8F   BB  257F  7197         PUSHR   #^M<R3,R5,R6,R7,R8,R10,R11>
                      50  D4  2583  7198         CLRL    R0                      ; Assume failure.
        53  0000215B'EF   DE  2585  7199         MOVAL   FORMAT_BUF,R3           ; Get address of temporary buffer.
                        258C  7200         CASE    FD$B_FORMAT(R10),-      ; Dispatch on format type.
                        258C  7201                 <ASCII_DISPLAY,-
                        258C  7202                 HEX_DISPLAY,-
                        258C  7203                 DEC_DISPLAY,-
                        258C  7204                 ASCIC_DISPLAY,-
                        258C  7205                 TIME_DISPLAY>,TYPE=B,LIMIT=#1
                        259B  7206
                        259B  7207
                        259B  7208 TIME_DISPLAY:
                        259B  7209         $ASCTIM_S -                     ; Convert time to ascii.
                        259B  7210                 TIMBUF=ASC_TIM_DSCR,-   ;
                        259B  7211                 TIMADR=(R7)             ;
                        25AE  7212 ;
                        25AE  7213 ; Shift the last tenth through eighteenth characters of the time string
                        25AE  7214 ; left two places.  This is to condense the date by effectively removing
                        25AE  7215 ; the first two digits of the year, i.e., dd-mmm-yyyy hh:mm:ss.cc becomes
                        25AE  7216 ; dd-mmm-yy hh:mmmm:ss.cc.  The last eigth characters are simply not output,
                        25AE  7217 ; leaving dd-mmm-yy hh:mm.
                        25AE  7218 ;
                        25AE  7219
        51  00001F9F'EF   DE  25AE  7220         MOVAL   TIMBUF+9,R1             ; Get address of "yyyy"
        57  00001F9D'EF   DE  25B5  7221         MOVAL   TIMBUF+7,R7             ; Get address of "yy h"
            87    81      7D  25BC  7222         MOVQ    (R1)+,(R7)+             ; Shift first four characters left 2.
            87    81      7D  25BF  7223         MOVQ    (R1)+,(R7)+             ; Shift next four characters left 2.
        57  00001F96'EF   DE  25C2  7224         MOVAL   TIMBUF,R7               ; Get address of time string.
                        25C9  7225                                         ; Fall through to ASCII_DISPLAY.
                        25C9  7226
                        25C9  7227
                        25C9  7228 ;
                        25C9  7229 ; Format the data in ASCII.
                        25C9  7230 ;
                        25C9  7231
                        25C9  7232 ASCII_DISPLAY:
                  56    D4  25C9  7233         CLRL    R6                      ; Zero index.
              04 AA     9A  25CB  7234         MOVZBL  FD$B_LENGTH(R10),-      ; Save length of string for return.
        00002153'EF         25CE  7235                 FORMAT_BUF_LEN          ;
          83   6746    90  25D3  7236 10$:     MOVB    (R7)[R6],(R3)+          ; Store a character.
                  56    96  25D7  7237         INCB    R6                      ; Bump index.
           04 AA  56    91  25D9  7238         CMPB    R6,FD$B_LENGTH(R10)     ; Are we done?
                  F4    12  25DD  7239         BNEQU   10$                     ; Loop if not.
              20  73    91  25DF  7240 20$:     CMPB    -(R3),#^A/ /            ; Any trailing spaces?
                  07    12  25E2  7241         BNEQU   30$                     ; Branch if not.
        F4 00002153'EF   F5  25E4  7242         SOBGTR  FORMAT_BUF_LEN,20$      ; Remove trailing space.
                  53    D6  25EB  7243 30$:     INCL    R3                      ; Restore R3 to last character plus 1.
                00DF    30  25ED  7244         BSBW    CHECK_VALID             ; Make sure the string contains valid
                        25F0  7245                                         ;  ascii characters.
                        25F0  7246         JLBC    R0,FERR                 ; Branch if string contains non-ascii.
                00CE    31  25F6  7247         BRW     FDONE                   ; Done.
```

```
                        25F9  7248
                        25F9  7249 ;
                        25F9  7250 ; Format the data in HEXADECIMAL
                        25F9  7251 ;
                        25F9  7252
                        25F9  7253 HEX_DISPLAY:
      56   04 AA   9A   25F9  7254          MOVZBL  FD$B_LENGTH(R10),R6     ; Get the number of bytes of data.
   5B   56   01   9C   25FD  7255          ROTL    #1,R6,R11               ; Get the maximum length of the
                        2601  7256                                          ;  converted string.
            56   D7   2601  7257 10$:     DECL    R6                      ; Count down number of bytes.
            0B   19   2603  7258          BLSS    15$                     ; Branch if data equals 0.
      58  6746   9A   2605  7259          MOVZBL  (R7)[R6],R8             ; Get a byte of data.
            16   12   2609  7260          BNEQ    20$                     ; Branch if not 2 leading zeros.
         5B   02   C2   260B  7261          SUBL    #2,R11                  ; Subtract 2 leading zeros from the
                        260E  7262                                          ;  length of the converted string.
            F1   11   260E  7263          BRB     10$                     ; Loop for all leading zeros.
00002153'EF   01   D0   2610  7264 15$:     MOVL    #1,FORMAT_BUF_LEN       ; Length equals 1.
0000215B'EF   30   90   2617  7265          MOVB    #^A/0/,FORMAT_BUF       ; String equals "0"
         00A6   31   261E  7266          BRW     FDONE                   ; Done.
58   58   FC 8F   78   2621  7267 20$:     ASHL    #-4,R8,R8               ; Scratch lower nyble.
            04   12   2626  7268          BNEQ    HIGH_NYBLE              ; Branch if not single leading zero.
            5B   D7   2628  7269          DECL    R11                     ; Subtract one leading zero from the
                        262A  7270                                          ;  length of the converted string.
            0C   11   262A  7271          BRB     LOW_NYBLE               ; Start format with low nyble.
                        262C  7272 HIGH_NYBLE:
      58  6746   9A   262C  7273          MOVZBL  (R7)[R6],R8             ; Get a byte of data.
58   58   FC 8F   9C   2630  7274          ROTL    #-4,R8,R8               ; Scratch the low nyble.
         00C4   30   2635  7275          BSBW    CONVERT_HEX             ; Convert the nyble to ascii.
                        2638  7276 LOW_NYBLE:
      58  6746   9A   2638  7277          MOVZBL  (R7)[R6],R8             ; Get full byte back.
58   000000F0 8F   CA   263C  7278          BICL    #^X000000F0,R8          ; Mask the low nyble.
         00B6   30   2643  7279          BSBW    CONVERT_HEX             ; Convert the nyble to ascii.
         E3 56   F4   2646  7280          SOBGEQ  R6,HIGH_NYBLE           ; Loop for entire value.
00002153'EF   5B   D0   2649  7281          MOVL    R11,FORMAT_BUF_LEN      ; Save length of conversion.
         0074   31   2650  7282          BRW     FDONE                   ; Done.
                        2653  7283
                        2653  7284
                        2653  7285 ;
                        2653  7286 ; Format the data in DECIMAL.
                        2653  7287 ;
                        2653  7288
                        2653  7289 DEC_DISPLAY:
      56   04 AA   9A   2653  7290          MOVZBL  FD$B_LENGTH(R10),R6     ; Get the number of bytes of data.
         08   56   91   2657  7291          CMPB    R6,#8                   ; We can't handle more than 8 bytes.
            6E   14   265A  7292          BGTR    FERR                    ; Error if more than 8 bytes.
            5A   7C   265C  7293          CLRQ    R10                     ; Start with quadword of zero.
            56   D7   265E  7294 10$:     DECL    R6                      ; Count down the bytes.
            0A   19   2660  7295          BLSS    15$                     ; Branch if we've got all the bytes.
   5A   5A   08   79   2662  7296          ASHQ    #8,R10,R10              ; Shift the byte into quadword.
      5A  6746   90   2666  7297          MOVB    (R7)[R6],R10            ; Get a byte of data.
            F2   11   266A  7298          BRB     10$                     ; Loop for entire value.
57   000004CA'EF   DE   266C  7299 15$:     MOVAL   DEC_TAB,R7              ; Point to conversion table.
         56   14   D0   2673  7300          MOVL    #DEC_TAB_LEN,R6         ; 20 conversions steps in all.
         58   87   7D   2676  7301 20$:     MOVQ    (R7)+,R8                ; Get a value from the conversion
                        2679  7302                                          ;  table.
         0098   30   2679  7303          BSBW    CONVERT_DEC             ; Convert to one decimal digit.
0000215B'8F   53   D1   267C  7304          CMPL    R3,#FORMAT_BUF          ; Have any characters been output yet?
```

```
                          04   12  2683  7305          BNEQU   30$                     ; Branch if so.
                          55   95  2685  7306          TSTB    R5                      ; Was the conversion result = 0.
                          04   13  2687  7307          BEQLU   40$                     ; If so, don't save the character.
                    83    55   30  2689  7308 30$:     ADDB3   #^X30,R5,(R3)+          ; Convert to ascii and save character.
                       E6 56   F5  268D  7309 40$:     SOBGTR  R6,20$                  ; Loop for all 20 steps.
00002153'EF     53   0000215B'8F  C3  2690  7310          SUBL3   #FORMAT_BUF,R3,-        ; Save the length of the conversion.
                                      269C  7311                  FORMAT_BUF_LEN          ;
                          0E   12  269C  7312          BNEQ    50$                     ; Branch if result wasn't zero.
                0000215B'EF    30  269E  7313          MOVB    #^A/0/,FORMAT_BUF       ; Store a "0"
                00002153'EF    01  26A5  7314          MOVL    #1,FORMAT_BUF_LEN       ; Length equals one byte.
                        0018   31  26AC  7315 50$:     BRW     FDONE                   ; Done.
                                   26AF  7316
                                   26AF  7317 ASCIC_DISPLAY:
                00002153'EF    87  26AF  7318          MOVZBL  (R7)+,FORMAT_BUF_LEN    ; Save length of string for return.
                          34   BB  26B6  7319          PUSHR   #^M<R2,R4,R5>
          63    67  00002153'EF   28  26B8  7320          MOVC3   FORMAT_BUF_LEN,(R7),(R3); Save the string.
                          34   BA  26C0  7321          POPR    #^M<R2,R4,R5>
                          0B   10  26C2  7322          BSB     CHECK_VALID             ; Make sure string contains valid ascii
                                   26C4  7323                                          ;  characters.
                       03 50   E9  26C4  7324          BLBC    R0,FERR                 ; Branch if not.
                                   26C7  7325
                       50   01   D0  26C7  7326 FDONE:  MOVL    #SS$_NORMAL,R0          ; Indicate success.
                       0DE8 8F   BA  26CA  7327 FERR:   POPR    #^M<R3,R5,R6,R7,R8,R10,R11>
                                 05  26CE  7328          RSB
                                     26CF  7329
```

```
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine checks the string in the format buffer for valid ascii
                        ;       characters.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB     CHECK_VALID
                        ;
                        ; CALLED BY:
                        ;
                        ;       ASCII_DISPLAY
                        ;       ASCIC_DISPLAY
                        ;
                        ; INPUTS:
                        ;
                        ;       FORMAT_BUF contains the string.
                        ;       FORMAT_BUF_LEN contains the length of the string.
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       R0 = SS$_NORMAL if string contains valid ascii characters.
                        ;
                        ; REGISTER USAGE:
                        ;
                        ;       R2 -    Counts down the characters in the string.
                        ;       R3 -    Points into the buffer to the character to check.
                        ;
                        ;--

                        CHECK_VALID:
                  0C BB         PUSHR   #^M<R2,R3>
                  50 D4         CLRL    R0                      ; Assume failure.
52  00002153'EF   D0            MOVL    FORMAT_BUF_LEN,R2       ; Get length of string.
53  0000215B'EF   DE            MOVAL   FORMAT_BUF,R3           ; Get address of format buffer.
                  03 10         BSB     CHECK_ASCII             ; Check for none ascii characters.
                  0C BA         POPR    #^M<R2,R3>
                     05         RSB
```

```
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Check a string for valid ascii characters.  Return immediately if an
                        ;       invalid ascii character is found with R3 pointing to the character and
                        ;       failed status in R0.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB     CHECK_ASCII
                        ;
                        ; CALLED BY:
                        ;
                        ;       CHECK_VALID
                        ;       BRDCST
                        ;
                        ; INPUTS:
                        ;
                        ;       R2 = Length of string
                        ;       R3 = Address of string
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       If all charcters in the string are valid ascii:
                        ;
                        ;               R0 = SS$_NORMAL
                        ;               R2 = 0
                        ;               R3 = pointer to character following the string.
                        ;
                        ;       If invalid character found:
                        ;
                        ;               R0 = failed status
                        ;               R2 = number of characters remaining in string.
                        ;               R3 = points to failed character.
                        ;
                        ;--
                        CHECK_ASCII:
50   D4                 CLRL    R0                      ; Assume failure.
20   63 91              CMPB    (R3),# A/ /             ; Less than ^X20 ?
     0E 19              BLSS    99$                     ; If so, character is not ascii.
7E 8F 63 91             CMPB    (R3),#^A/~/             ; Greater than ^X7E ?
     08 14              BGTR    99$                     ; If so, character is not ascii.
     53 D6              INCL    R3                      ; Update pointer.
   F0 52 F5             SOBGTR  R2,10$                  ; Loop for entire string.
50   01 D0              MOVL    #SS$_NORMAL,R0          ; Indicate success.
        05              RSB
                        99$:
```

```
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine converts the value in R8 to one ascii format hex digit
                        ;       and outputs the digit to the format buffer.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB     CONVERT_HEX
                        ;
                        ; CALLED BY:
                        ;
                        ;       LOW_NYBLE
                        ;       HIGH_NYBLE
                        ;
                        ; INPUTS:
                        ;
                        ;       R3 -    Points to the position in the format buffer where the result
                        ;               will be stored.
                        ;       R8 -    Contains the value to be converted in the low nyble.
                        ;
                        ; OUTPUTS:
                        ;
                        ;       One ascii character output to the format buffer.
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       R3 updated.
                        ;       R8 scratched.
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       None.
                        ;
                        ;--

                        CONVERT_HEX:
      0A   58   91      CMPB    R8,#10                  ; Value less than 10?
           05   1E      BGEQU   10$                     ; Branch if not.
      58   30   80      ADDB    #^X30,R8                ; Convert value 0 through 9 to ascii.
           0A   11      BRB     20$                     ; Jump to common code.
      58   0A   C2      SUBL    #10,R8                  ; Convert value ^X0A through ^X0F to
58 00000041   8F   C0   ADDL    #^X41,R8                ;  ascii.
      83   58   90      MOVB    R8,(R3)+                ; Store the converted character.
                05      RSB
```

```
                     2714  7474 ;++
                     2714  7475 ;
                     2714  7476 ; FUNCTIONAL DESCRIPTION:
                     2714  7477 ;
                     2714  7478 ;       This routine performs one step of hex to decimal conversion by dividing
                     2714  7479 ;       the quadword in R10 by the quadword in R8.  R8 contains the decimal
                     2714  7480 ;       weight of the digit for this step of the conversion.
                     2714  7481 ;
                     2714  7482 ; CALLING SEQUENCE:
                     2714  7483 ;
                     2714  7484 ;       JSB     CONVERT_DEC
                     2714  7485 ;
                     2714  7486 ; CALLED BY:
                     2714  7487 ;
                     2714  7488 ;       DEC_DISPLAY
                     2714  7489 ;
                     2714  7490 ; INPUTS:
                     2714  7491 ;
                     2714  7492 ;       R8  -   Quadword value for this step from the conversion table.
                     2714  7493 ;       R10 -   Quadword value to be converted.
                     2714  7494 ;
                     2714  7495 ; OUTPUTS:
                     2714  7496 ;
                     2714  7497 ;       R5 -    Decimal digit resulting from this step of the conversion.
                     2714  7498 ;       R8 -    Remainder to be used in next step of conversion.
                     2714  7499 ;
                     2714  7500 ; RETURN STATUS:
                     2714  7501 ;
                     2714  7502 ;       None.
                     2714  7503 ;
                     2714  7504 ;--
                     2714  7505
                     2714  7506 CONVERT_DEC:
     55    94        2714  7507          CLRB    R5                      ; Assume result equals zero.
5A   58    C2        2716  7508 10$:     SUBL    R8,R10                  ; Quadword subtract with carry.
5B   59    D9        2719  7509          SBWC    R9,R11                  ;
     04    1F        271C  7510          BCS     20$                     ; Jump if less than zero.
     55    96        271E  7511          INCB    R5                      ; Otherwise result = result + 1.
     F4    11        2720  7512          BRB     10$                     ; Loop.
5A   58    C0        2722  7513 20$:     ADDL    R8,R10                  ; Quadword add to restore remainder.
5B   59    D8        2725  7514          ADWC    R9,R11                  ;
               05    2728  7515          RSB
                     2729  7516
                     2729  7517
```

```
2729  7519          .SBTTL  Data display routines.
2729  7520 ;
2729  7521 ; Addresses of data structures are passed to the data display routines in a
2729  7522 ; call frame.  All addresses are determined by the routines CLUSTER_REPORT
2729  7523 ; and LOCAL_REPORT, which determine the format of the report.
2729  7524 ;
2729  7525 ;
2729  7526 ;                    +-------------------------------+
2729  7527 ;         (AP)       |      Number of elements       |
2729  7528 ;                    +-------------------------------+
2729  7529 ;  (AP)[CLUB_IN]    |    Base address of CLUB       |
2729  7530 ;                    +-------------------------------+
2729  7531 ;    (AP)[SB_IN]    | Base address of system block  |
2729  7532 ;                    +-------------------------------+
2729  7533 ;   (AP)[CSB_IN]    |    Base address of CSB        |
2729  7534 ;                    +-------------------------------+
2729  7535 ;    (AP)[PB_IN]    |  Base address of path block   |
2729  7536 ;                    +-------------------------------+
2729  7537 ;   (AP)[CDT_IN]    |    Base address of CDT        |
2729  7538 ;                    +-------------------------------+
2729  7539 ;   (AP)[PDT_IN]    |    Base address of PDT        |
2729  7540 ;                    +-------------------------------+
2729  7541 ;    (AP)[MB_IN]    | Address of miscellaneous buffer|
2729  7542 ;                    +-------------------------------+
2729  7543 ;
2729  7544
2729  7545
```

```
                     2729  7547          .SBTTL  -       GET_DATA routine.
                     2729  7548 ;++
                     2729  7549 ;
                     2729  7550 ; FUNCTIONAL DESCRIPTION:
                     2729  7551 ;
                     2729  7552 ;       This routine gets a line of data into the data buffer by calling a
                     2729  7553 ;       routines that extract and format the data for each of the fields that
                     2729  7554 ;       are to be displayed.  The data is then stored in DATA_BUF for output.
                     2729  7555 ;
                     2729  7556 ; CALLING SEQUENCE:
                     2729  7557 ;
                     2729  7558 ;       CALLS   GET_DATA
                     2729  7559 ;
                     2729  7560 ; INPUTS:
                     2729  7561 ;
                     2729  7562 ;       Addresses of data structures from which the data is to be extracted are
                     2729  7563 ;       stored in a call frame.  See above picture for details.
                     2729  7564 ;
                     2729  7565 ; IMPLICIT INPUTS:
                     2729  7566 ;
                     2729  7567 ;       ADRS_TABLE, BITMAP_ADRS, and MAP_LEN must be set up according to the
                     2729  7568 ;       FIND_ENTRY routine.
                     2729  7569 ;
                     2729  7570 ; OUTPUTS:
                     2729  7571 ;
                     2729  7572 ;       One line of ascii formatted data stored in DATA_BUF.
                     2729  7573 ;
                     2729  7574 ; RETURN STATUS:
                     2729  7575 ;
                     2729  7576 ;       R0 =    SS$_NORMAL.
                     2729  7577 ;
                     2729  7578 ; REGISTER USAGE:
                     2729  7579 ;
                     2729  7580 ;       R3 -    Pointer to next free character position in data buffer.
                     2729  7581 ;       R6 -    Counts down the number of characters that can be added
                     2729  7582 ;               to the data buffer before its length exceeds the width
                     2729  7583 ;               of the screen.
                     2729  7584 ;       R7 -    Contains the offset into the data structure used to extract
                     2729  7585 ;               the data.
                     2729  7586 ;       R10 -   Points to field descriptor.
                     2729  7587 ;       R11 -   Used to determine the number of fill characters (asterisks,
                     2729  7588 ;               question marks, or spaces) to add to the display field.
                     2729  7589 ;
                     2729  7590 ;--
                     2729  7591
                 0EC8  2729  7592 .ENTRY  GET_DATA,^M<R3,R6,R7,R9,R10,R11>
00000476'EF    04    CA  272B  7593          BICL    #MF$M_INVERSE,MISC_FLAGS; Make sure inverse flags starts out
                     2732  7594                                          ;  clear.
                  00   E1  2732  7595          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
       2C 00000000'EF       2734  7596                  CURRENT_REPORT,3$       ;  cluster report.
17 00000476'EF    04   E1  273A  7597          BBC     #MF$V_CONT,MISC_FLAGS,2$; Branch if noncontinuous display.
0000225C'EF 00002248'EF    D1  2742  7598          CMPL    LINE_NUM,CMD_LINE       ; Have we overflowed the screen?
                  0A   19  274D  7599          BLSS    2$                      ; Branch if not.
                  18   88  274F  7600          BISB    #SCR$M_MORE_DWN!SCR$M_MORE,-
          0000047A'EF       2751  7601                  SCROLL_FLAGS            ; Indicate screen can be scrolled up.
                022C   31  2756  7602          BRW     50$                     ; No need to get data for the line.
          00000464'EF   D0  2759  7603 2$:     MOVL    CLSTR_H_SCROLL,-        ; Get a copy of the scroll value for the
```

```
                00000046C'EF      275F  7604                  H_SCROLL_LINE           ; cluster report.
                          0B  11  2764  7605          BRB     4$                      ; Branch to common code.
                00000468'EF  D0   2766  7606 3$:      MOVL    LOCAL_H_SCROLL,-        ; Get a copy of the scroll value for the
                00000046C'EF      276C  7607                  H_SCROLL_LINE           ; local report.
                00000460'EF  D7   2771  7608 4$:      DECL    V_SCROLL_LINE           ; Count down the number of lines of
                                  2777  7609                                          ; data that are scrolled off the
                                  2777  7610                                          ; screen.
                                  2777  7611          JGEQ    5O$                     ; Skip this line if it's scrolled off
                                  277C  7612                                          ; the screen.
          53    000019A3'EF  DE   277C  7613          MOVAL   DATA_BUF,R3             ; Get pointer to data buffer.
          56    00002222'EF  3C   2783  7614          MOVZWL  SCREEN_WIDTH,R6         ; Get screen width.
                          01  E0  278A  7615          BBS     #SCR$V_MORE_LFT,-       ; Branch if screen has been scrolled
            08 0000047A'EF        278C  7616                  SCROLL_FLAGS,5$         ; left.
                        FC8A  30  2792  7617          BSBW    DELM                    ; Add starting field delimiter to line.
                                  2795  7618          JLEQU   40$                     ; Branch if it won't fit on the screen.
                00002270'EF  D4   279A  7619 5$:      CLRL    START_BIT               ; Start scanning the bitmap with bit
                                  27A0  7620                                          ; zero.
                          00  E1  27A0  7621          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
            09 00000000'EF        27A2  7622                  CURRENT_REPORT,6$       ; cluster report.
      0000226C'EF       02  D0  27A8  7623          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Save length of cluster bitmap.
                          07  11  27AF  7624          BRB     10$                     ; Jump to common code.
      0000226C'EF       00  D0  27B1  7625 6$:       MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Save length of local bitmap.
                      FBA8  30  27B8  7626 10$:      BSBW    FIND_ENTRY              ; Find an entry.
                                27BB  7627          JLBC    R0,40$                  ; Branch if an entry wasn't found.
                     0C AA  D3  27C1  7628          BITL    FD$L_CLASS(R10),-       ; Is this entry in a different class?
                00000001'EF       27C4  7629                  CLASS_SELECT            ;
                          ED  13  27C9  7630          BEQLU   10$                     ; Branch if so.
                     0C AA  D3  27CB  7631          BITL    FD$L_CLASS(R10),-       ; Is this class enabled?
                000021F8'EF       27CE  7632                  DATA_CNTRL              ;
                          11  12  27D3  7633          BNEQU   20$                     ; Branch if so.
                                27D5  7634 ;
                                27D5  7635 ; Class is disabled, so fill this field with spaces.
                                27D5  7636 ;
                        FC6B  30  27D5  7637          BSBW    ADD_ONE_SPACE           ; Preceed the data with atleast one
                                27D8  7638                                          ; space.
                                27D8  7639          JLEQU   40$                     ; Branch if it won't fit.
                   59   6A  9A  27DD  7640          MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of the field.
                      FC66  30  27E0  7641          BSBW    ADD_SPACES              ; Fill the field with spaces.
                      0106  31  27E3  7642          BRW     29$                     ; Jump to common code.
                                27E6  7643
                    57   10 AA  D0  27E6  7644 20$:      MOVL    FD$L_OFFSET(R10),R7     ; Get the SCS offset.
                      08 BA  16  27EA  7645          JSB     @FD$C_DISPLAY_ROUTINE(R10)
                                27ED  7646                                          ; Call the display routine to extract
                                27ED  7647                                          ; and format the data.
                   59   50  D0  27ED  7648          MOVL    R0,R9                   ; Save status from display routine.
                          02  E1  27F0  7649          BBC     #MF$V_INVERSE,-         ; Branch if the data hasn't changed
            20 00000476'EF        27F2  7650                  MISC_FLAGS,22$          ; since the last snapshot.
00 00000434'EF    00002248'EF  E2  27F8  7651          BBSS    LINE_NUM,LINE_DIF,21$   ; Indicate the some data in this line
                                2804  7652                                          ; changed and the line must be
                                2804  7653                                          ; rewritten to the screen.
         0C 00002247'EF       01  E1  2804  7654 21$:      BBC     #DVT$V_DCRT,OUT_DEV,22$ ; Branch if not DECcrt terminal.
                00000017E'EF  28  280C  7655          MOVC3   INVRS_DSCR,-            ; Store sequence to turn on inverse
                   63   00000186'EF       2812  7656                  INVRS_DSCR+8,(R3)       ; video.
                        FC28  30  2818  7657 22$:      BSBW    ADD_ONE_SPACE           ; Preceed data with one space.
                                281B  7658                                          ; Branch to display subroutine for this
                                281B  7659                                          ; field.
                                281B  7660          JLEQU   40$                     ; Branch if it didn't fit.
```

```
                 2E 59   E8  2820  7661          BLBS    R9,25$                  ; Branch if display routine succeeded.
                             2823  7662 ;
                             2823  7663 ; Display routine failed, so center a question mark in the field.
                             2823  7664 ;
                     59  6A   9A  2823  7665          MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of this field.
                         59   97  2826  7666          DECB    R9                      ; Subtract one for the question mark.
              59     59   FF 8F   9C  2828  7667          ROTL    #-1,R9,R9               ; Calculate the number of spaces needed
                             282D  7668                                                  ;  to center the question mark.
                       FC19   30  282D  7669          BSBW    ADD_SPACES              ; Add leading spaces to buffer.
                             2830  7670          JLEQU   40$                     ; Branch if the spaces won't fit on the
                             2835  7671                                                  ;  screen.
                 08 59   1F   E1  2835  7672          BBC     #31,R9,23$              ; Branch if even number of spaces.
                       FC07   30  2839  7673          BSBW    ADD_ONE_SPACE           ; Add one more leading space.
                             283C  7674          JLEQU   40$                     ; Branch if it didn't fit.
                     83  3F   90  2841  7675 23$:     MOVB    #^A/?/,(R3)+            ; Store the question mark.
                         56   D7  2844  7676          DECL    R6                      ; Update number of columns remaining.
                             2846  7677          JLEQU   40$                     ; Branch if the question mark didn't
                             284B  7678                                                  ;  fit on the screen.
                       FBFB   30  284B  7679          BSBW    ADD_SPACES              ; Add trailing spaces to buffer.
                       009B   31  284E  7680          BRW     29$                     ; Jump to common code.
                             2851  7681
                     59  6A   9A  2851  7682 25$:     MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of this field.
              59   00002153'EF   D1  2854  7683          CMPL    FORMAT_BUF_LEN,R9       ; Will the formatted data fit in the
                             285B  7684                                                  ;  display field?
                         78   13  285B  7685          BEQLU   28$                     ; Branch if it fits exactly.
                         24   1F  285D  7686          BLSSU   27$                     ; Branch if its smaller.
                   05 AA   91  285F  7687          CMPB    FD$B_FORMAT(R10),-      ; Is the data formatted in hex?
                         02        2862  7688                  #HEX                    ;
                         0F   13  2863  7689          BEQLU   26$                     ; Branch if so.
                   05 AA   91  2865  7690          CMPB    FD$B_FORMAT(R10),-      ; Is the data formatted in decimal?
                         03        2868  7691                  #DEC                    ;
                         09   13  2869  7692          BEQLU   26$                     ; Branch if so.
                             286B  7693 ;
                             286B  7694 ; The data is ascii, but is larger than the field, so truncate the string.
                             286B  7695 ;
                             286B  7696
          00002153'EF    59   D0  286B  7697          MOVL    R9,FORMAT_BUF_LEN       ; Truncate the string by making its
                             2872  7698                                                  ;  length equal to the field size.
                         61   11  2872  7699          BRB     28$                     ; Copy the string.
                             2874  7700
                             2874  7701 ;
                             2874  7702 ; The data is non ascii and is larger than the field, so fill the field with
                             2874  7703 ; asterisks.
                             2874  7704 ;
                             2874  7705
                         34   BB  2874  7706 26$:     PUSHR   #^M<R2,R4,R5>
 63   59   2A   63   00   2C  2876  7707          MOVC5   #0,(R3),#^A/*/,R9,(R3)  ; Fill field with asterisks.
                         34   BA  287C  7708          POPR    #^M<R2,R4,R5>
                   56    59   C2  287E  7709          SUBL    R9,R6                   ; See if it fit on the screen.
                         69   11  2881  7710          BRB     29$                     ; Jump to common code.
                             2883  7711 ;
                             2883  7712 ; If the string is ascii then left justify the string in the field.  If
                             2883  7713 ; hex or decimal then right justify the string.
                             2883  7714 ;
                             2883  7715
                             2883  7716 27$:
          59   00002153'EF    C2  2883  7717          SUBL    FORMAT_BUF_LEN,R9       ; Get number of spaces to add.
```

```
                02   05 AA   91   288A  7718          CMPB    FD$B_FORMAT(R10),#HEX   ; Is the data to be displayed in
                                  288E  7719                                          ;  hexadecimal?
                       24   13   288E  7720          BEQL    275$                    ; Branch if so.
                03   05 AA   91   2890  7721          CMPB    FD$B_FORMAT(R10),#DEC   ; Is the data to be displayed in
                                  2894  7722                                          ;  decimal?
                       1E   13   2894  7723          BEQL    275$                    ; Branch if so.
                       34   BB   2896  7724          PUSHR   #^M<R2,R4,R5>
                00002153'EF 28   2898  7725          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
     63         0000215B'EF      289E  7726                  FORMAT_BUF,(R3)         ;
                       34   BA   28A4  7727          POPR    #^M<R2,R4,R5>
     56         00002153'EF C2   28A6  7728          SUBL    FORMAT_BUF_LEN,R6       ; Calculate columns remaining.
                       68   1B   28AD  7729          BLEQU   40$                     ; Branch if the string didn't fit on
                                  28AF  7730                                          ;  the screen.
                     FB97   30   28AF  7731          BSBW    ADD_SPACES              ; Add trailing spaces.
                       38   11   28B2  7732          BRB     29$                     ; Branch to common code.
                                  28B4  7733 275$:
                     FB92   30   28B4  7734          BSBW    ADD_SPACES              ; Add leading spaces.
                                  28B7  7735          JLEQU   40$                     ; Branch if the spaces didn't fit on
                                  28BC  7736                                          ;  the screen.
                       34   BB   28BC  7737          PUSHR   #^M<R2,R4,R5>
                00002153'EF 28   28BE  7738          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
     63         0000215B'EF      28C4  7739                  FORMAT_BUF,(R3)         ;
                       34   BA   28CA  7740          POPR    #^M<R2,R4,R5>
     56         00002153'EF C2   28CC  7741          SUBL    FORMAT_BUF_LEN,R6       ; Calculate columns remaining.
                       17   11   28D3  7742          BRB     29$                     ; Branch to common code.
                       34   BB   28D5  7743 28$:     PUSHR   #^M<R2,R4,R5>
                00002153'EF 28   28D7  7744          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
     63         0000215B'EF      28DD  7745                  FORMAT_BUF,(R3)         ;
                       34   BA   28E3  7746          POPR    #^M<R2,R4,R5>
     56         00002153'EF C2   28E5  7747          SUBL    FORMAT_BUF_LEN,R6       ; Will it fit on the screen?
                       29   1B   28EC  7748 29$:     BLEQU   40$                     ; Branch if it didn't fit.
                     FB52   30   28EE  7749          BSBW    ADD_ONE_SPACE           ; Follow with one space.
                       24   1B   28F1  7750          BLEQU   40$                     ; Branch if it didn't fit.
                       02   E5   28F3  7751          BBCC    #MF$V_INVERSE,-         ; Branch if the data hasn't changed
          14 00000476'EF        28F5  7752                  MISC_FLAGS,30$          ;  since the last snapshot.
     0C 00002247'EF   01   E1   28FB  7753          BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if not DECcrt terminal.
                00000143'EF 28   2903  7754          MOVC3   DCRT_NOATR_DSCR,-       ; Store sequence to turn off attributes
          63    0000014B'EF      2909  7755                  DCRT_NOATR_DSCR+8,(R3)  ;  (inverse video).
                     FB0D   30   290F  7756 30$:     BSBW    DELM                    ; Add ending delimiter to line.
                                  2912  7757          JGTRU   10$                     ; Loop if there's more room on the line.
                       02   E5   2917  7758 40$:     BBCC    #MF$V_INVERSE,-         ; Branch if the data hasn't changed
          14 00000476'EF        2919  7759                  MISC_FLAGS,45$          ;  since the last snapshot.
     0C 00002247'EF   01   E1   291F  7760          BBC     #DVT$V_DCRT,OUT_DEV,45$ ; Branch if not DECcrt terminal.
                00000143'EF 28   2927  7761          MOVC3   DCRT_NOATR_DSCR,-       ; Store sequence to turn off attributes
          63    0000014B'EF      292D  7762                  DCRT_NOATR_DSCR+8,(R3)  ;  (inverse video).
0000199B'CF    53   000019A3'8F C3   2933  7763 45$:     SUBL3   #DATA_BUF,R3,DATA_DSCR  ; Save length of data line.
                5B   0000199B'EF DE   293F  7764          MOVAL   DATA_DSCR,R11           ; Get address of descriptor for the
                                  2946  7765                                          ;  line of data.
                                  2946  7766
          34 00000476'EF   03   E0   2946  7767          BBS     #MF$V_RIP,MISC_FLAGS,49$; If refresh in progress, make sure
                                  294E  7768                                          ;  we draw all the lines.
  28 0000043C'EF   00002248'EF E0   294E  7769          BBS     LINE_NUM,BLANK_LINES,49$; Branch if this is a new blank line.
  1C 00000434'EF   00002248'EF E0   295A  7770          BBS     LINE_NUM,LINE_DIF,49$   ; Branch if this line has changed.
     0C 00002247'EF   01   E1   2966  7771          BBC     #DVT$V_DCRT,OUT_DEV,48$ ; Branch if output device is not a
                                  296E  7772                                          ;  DECcrt terminal.
                00002248'EF E0   296E  7773          BBS     LINE_NUM,-              ; Branch if this line changed on the
          08 00000438'EF         2974  7774                  PRIOR_LINE_DIF,49$      ;  last pass and must be rewritten to
```

```
                     297A  7775
                     297A  7776 48$:    INCL    LINE_NUM                ; remove the inverse video.
00002248'EF   D6     297A  7776                                         ; Otherwise, just bump the line number.
          03   11    2980  7777         BRB     50$                     ; And finish up.
        FAE1   30    2982  7778 49$:    BSBW    DRAW_LINE               ; Output the line.
     50   01   D0    2985  7779 50$:    MOVL    #SS$_NORMAL,R0
               04    2988  7780         RET
                     2989  7781
                     2989  7782
```

```
                     2989  7784          .SBTTL  -       General data display routines.
                     2989  7785 ;++
                     2989  7786 ;
                     2989  7787 ; FUNCTIONAL DESCRIPTION:
                     2989  7788 ;
                     2989  7789 ;       This is the default data display routine.  It gets the address of the
                     2989  7790 ;       data by first determining the type of structure containing the data
                     2989  7791 ;       by examining the prefix of the structure name argument.  If the prefix
                     2989  7792 ;       is valid, it is converted to an index into the call frame which
                     2989  7793 ;       contains the base address of the data structures containing the data.
                     2989  7794 ;       the base address of the proper structure is then added to R7 to get
                     2989  7795 ;       the address of the data. FORMAT_DATA is then called to format the data.
                     2989  7796 ;
                     2989  7797 ; Note: DISPLAY_FIELD can only be used when the data to be displayed is found
                     2989  7798 ;       in either the club, a cluster system block, an SCS system block, path
                     2989  7799 ;       block, or connection descriptor table, a port descriptor table, or the
                     2989  7800 ;       miscellaneous buffer.  The argument STRUCTURE_NAME given in the
                     2989  7801 ;       DEFINE_FIELD macro must be a valid SCS offset and must begin with CLUB.
                     2989  7802 ;       SB$, PB$, CDT$, PDT$ or a miscellaneous buffer offset begining with MB$.
                     2989  7803 ;       Any other argument will cause the field to be filled with question
                     2989  7804 ;       marks.
                     2989  7805 ;
                     2989  7806 ;
                     2989  7807 ; CALLING SEQUENCE:
                     2989  7808 ;
                     2989  7809 ;       JSB     DISPLAY_FIELD
                     2989  7810 ;
                     2989  7811 ; CALLED BY:
                     2989  7812 ;
                     2989  7813 ;       GET_DATA
                     2989  7814 ;
                     2989  7815 ; INPUTS:
                     2989  7816 ;
                     2989  7817 ;       R7      = SCS offset from DEFINE_FIELD macro.
                     2989  7818 ;       R10     = Address of field descriptor.
                     2989  7819 ;
                     2989  7820 ;
                     2989  7821 ; IMPLICIT INPUTS:
                     2989  7822 ;
                     2989  7823 ;       AP      = Address of call frame containing the addresses of the data
                     2989  7824 ;                 structures to be used when data is extracted.
                     2989  7825 ;
                     2989  7826 ; OUTPUTS:
                     2989  7827 ;
                     2989  7828 ;       Formated data string returned in FORMAT_BUF.
                     2989  7829 ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                     2989  7830 ;
                     2989  7831 ;
                     2989  7832 ; RETURN STATUS:
                     2989  7833 ;
                     2989  7834 ;       R0 =    SS$_NORMAL or failed, or status from FORMAT_DATA.
                     2989  7835 ;
                     2989  7836 ;--
                     2989  7837
                     2989  7838 DISPLAY_FIELD:
01FC 8F    BB        2989  7839          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>
     50    D4        298D  7840          CLRL    R0                      ; Assume failure.
```

```
                58   01    D0   298F  7841           MOVL    #1,R8                       ; Set index past number of arguments in
                                 2992  7842                                              ;  call frame.
      54  00000489'EF    DE   2992  7843           MOVAL   SYM_PRFX,R4                 ; Point to array of symbol prefixes.
          55    14 AA    DE   2999  7844           MOVAL   FD$C_OFFSET_PRFX(R10),R5    ; Get the address of the src2 string.
                53   84    9A   299D  7845 10$:     MOVZBL  (R4)+,R3                    ; Get the length of src1 string
                     08    BB   29A0  7846           PUSHR   #^M<R3>
          65    64   53    29   29A2  7847           CMPC    R3,(R4),(R5)                ; Check for valid prefix.
                     08    BA   29A6  7848           POPR    #^M<R3>
                     0B    13   29A8  7849           BEQL    20$                         ; Branch if valid prefix.
                54   53    C0   29AA  7850           ADDL    R3,R4                       ; Point to next prefix.
    FFEA 58     01   56    F1   29AD  7851           ACBL    R6,#1,R8,10$                ; Loop for all arguments.
                     2C    11   29B3  7852           BRB     99$                         ; Branch if invalid prefix.
      54  0000049C'8F    D1   29B5  7853 20$:     CMPL    #CDT_PFX+1,R4               ; Is the prefix "CDT$"?
                     09    12   29BC  7854           BNEQ    30$                         ; Branch if not.
      57  FFFFFFF0 8F    C0   29BE  7855           ADDL    #SHC$L_CDT_DIF,R7           ; Adjust CDT offset to compensate for
                                 29C5  7856                                              ;  portion of CDT not stored.
                     10    11   29C5  7857           BRB     40$                         ; Skip check for "PDT$".
      54  000004A1'8F    D1   29C7  7858 30$:     CMPL    #PDT_PFX+1,R4               ; Is the prefix "PDT$"?
                     07    12   29CE  7859           BNEQ    40$                         ; Branch if not.
      57  FFFFFF4C 8F    C0   29D0  7860           ADDL    #SHC$L_PDT_DIF,R7           ; Adjust PDT offset to compensate for
                                 29D7  7861                                              ;  portion of PDT not stored.
            57   6C48    C0   29D7  7862 40$:     ADDL    (R12)[R8],R7                ; Get address of data.
                 035F    30   29DB  7863           BSBW    CHECK_DIF                   ; Check for change in data since last
                                 29DE  7864                                              ;  scan.
                 FB9E    30   29DE  7865           BSBW    FORMAT_DATA                 ; Format the data.
              01FC 8F    BA   29E1  7866 99$:     POPR    #^M<R2,R3,R4,R5,R6,R7,R8>
                         05   29E5  7867           RSB
                              29E6  7868
                              29E6  7869
```

```
                     29E6  7871 ;++
                     29E6  7872 ;
                     29E6  7873 ; FUNCTIONAL DESCRIPTION:
                     29E6  7874 ;
                     29E6  7875 ;       This routine is used to convert a numeric value into a string.  The
                     29E6  7876 ;       value is used as an index into a list of counted ascii strings.  The
                     29E6  7877 ;       address of the indexed string is then used when calling the FORMAT_DATA
                     29E6  7878 ;       routine.
                     29E6  7879 ;
                     29E6  7880 ; Note: The FORMAT argument in the field descriptor for any field using this
                     29E6  7881 ;       routine must be of the type "ASCIC".
                     29E6  7882 ;
                     29E6  7883 ; CALLING SEQUENCE:
                     29E6  7884 ;
                     29E6  7885 ;       JSB     DISPLAY_STRING
                     29E6  7886 ;
                     29E6  7887 ; CALLED BY:
                     29E6  7888 ;
                     29E6  7889 ;       DISPLAY_RPORT_TYPE
                     29E6  7890 ;       DISPLAY_CABLE_STATUS
                     29E6  7891 ;       DISPLAY_LOOPBACK_STATUS
                     29E6  7892 ;
                     29E6  7893 ; INPUTS:
                     29E6  7894 ;
                     29E6  7895 ;       R7      = Value to be used as an index into a list of counted ascii
                     29E6  7896 ;                 strings.
                     29E6  7897 ;       R8      = Address of a list of counted ascii strings.
                     29E6  7898 ;
                     29E6  7899 ; OUTPUTS:
                     29E6  7900 ;
                     29E6  7901 ;       Formated data string returned in FORMAT_BUF.
                     29E6  7902 ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                     29E6  7903 ;
                     29E6  7904 ;
                     29E6  7905 ; RETURN STATUS:
                     29E6  7906 ;
                     29E6  7907 ;       R0 =    SS$_NORMAL or failed, or status from FORMAT_DATA.
                     29E6  7908 ;
                     29E6  7909 ;--
                     29E6  7910
                     29E6  7911 DISPLAY_STRING:
0380 8F    BB        29E6  7912          PUSHR   #^M<R7,R8,R9>
     57    D5        29EA  7913          TSTL    R7                      ; Index equal zero?
     09    13        29EC  7914          BEQLU   20$                     ; Branch if so.
59   88    9A        29EE  7915 10$:     MOVZBL  (R8)+,R9                ; Get the number of characters in the
                     29F1  7916                                          ;  descriptor.
58   59    C0        29F1  7917          ADDL    R9,R8                   ; Point to next descriptor.
  F7 57    F5        29F4  7918          SOBGTR  R7,10$                  ; Loop till R7 equals 0 and R8 points
                     29F7  7919                                          ;  to the string.
57   58    C0        29F7  7920 20$:     ADDL    R8,R7                   ; Get address of string.
   FB82    30        29FA  7921          BSBW    FORMAT_DATA             ; Format the string.
0380 8F    BA        29FD  7922          POPR    #^M<R7,R8,R9>
     05    05        2A01  7923          RSB
                     2A02  7924
                     2A02  7925
```

```
                     2A02  7927          .SBTTL  -       Cluster class display routines.
                     2A02  7928
                     2A02  7929 ;++
                     2A02  7930 ;
                     2A02  7931 ; FUNCTIONAL DESCRIPTION:
                     2A02  7932 ;
                     2A02  7933 ;       The following routines are used to display the bit in the CLUB$L_FLAGS
                     2A02  7934 ;       longword in the CLUB and the CSB$L_STATUS longword in the CSB.  These
                     2A02  7935 ;       bits are displayed as a one or zero only.
                     2A02  7936 ;
                     2A02  7937 ; CALLING SEQUENCE:
                     2A02  7938 ;
                     2A02  7939 ;       JSB     DISPLAY_QF_VOTE
                     2A02  7940 ;       JSB     DISPLAY_QF_SAME
                     2A02  7941 ;       JSB     DISPLAY_QF_ACTIVE
                     2A02  7942 ;
                     2A02  7943 ; CALLED BY:
                     2A02  7944 ;
                     2A02  7945 ;       GET_DATA
                     2A02  7946 ;
                     2A02  7947 ; INPUTS:
                     2A02  7948 ;
                     2A02  7949 ;       R7      = Offset of longword containing the bit to display.
                     2A02  7950 ;       (AP)[CLUB_IN]   = Base address of club or cluster system block.
                     2A02  7951 ;
                     2A02  7952 ; IMPLICIT INPUTS:
                     2A02  7953 ;
                     2A02  7954 ;       AP      = Address of call frame containing the addresses of the data
                     2A02  7955 ;                 structures to be used when data is extracted.
                     2A02  7956 ;
                     2A02  7957 ; OUTPUTS:
                     2A02  7958 ;
                     2A02  7959 ;       Formated data string returned in FORMAT_BUF.
                     2A02  7960 ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                     2A02  7961 ;
                     2A02  7962 ; REGISTER USAGE:
                     2A02  7963 ;
                     2A02  7964 ;       R7      = Offset into data structure where data is located.  Converted
                     2A02  7965 ;                 into actual address of data.
                     2A02  7966 ;       R8      = Index into call frame to locate structure base address.
                     2A02  7967 ;--
                     2A02  7968
                     2A02  7969 DISPLAY_QF_VOTE:
  01C0 8F    BB      2A02  7970          PUSHR   #^M<R6,R7,R8>
  58   01    D0      2A06  7971          MOVL    #CLUB_IN,R8                 ; Set index into call frame to club
                     2A09  7972                                              ;  base address.
57   6C48    C0      2A09  7973          ADDL    (R12)[R8],R7                ; Get address of data.
  56   19    D0      2A0D  7974          MOVL    #CLUB$V_QF_VOTE,R6          ; Get number of bit to check.
     0031    30      2A10  7975          BSBW    DISPLAY_QF                  ; Convert to formatted string.
  01C0 8F    BA      2A13  7976          POPR    #^M<R6,R7,R8>
             05      2A17  7977          RSB
                     2A18  7978
                     2A18  7979 DISPLAY_QF_SAME:
  01C0 8F    BB      2A18  7980          PUSHR   #^M<R6,R7,R8>
  58   03    D0      2A1C  7981          MOVL    #CSB_IN,R8                  ; Set index into call frame to cluster
                     2A1F  7982                                              ;  system block base address.
57   6C48    C0      2A1F  7983          ADDL    (R12)[R8],R7                ; Get address of data.
```

```
                56   03   D0  2A23  7984          MOVL    #CSB$V_QF_SAME,R6       ; Get number of bit to check.
                   001B   30  2A26  7985          BSBW    DISPLAY_QF              ; Convert to formatted string.
                01C0 8F   BA  2A29  7986          POPR    #^M<R6,R7,R8>
                          05  2A2D  7987          RSB
                              2A2E  7988
                              2A2E  7989 DISPLAY_QF_ACTIVE:
                01C0 8F   BB  2A2E  7990          PUSHR   #^M<R6,R7,R8>
                58   03   D0  2A32  7991          MOVL    #CSB_IN,R8              ; Set index into call frame to cluster
                              2A35  7992                                          ;  system block base address.
              57   6C48   C0  2A35  7993          ADDL    (R12)[R8],R7            ; Get address of data.
                56   09   D0  2A39  7994          MOVL    #CSB$V_QF_ACTIVE,R6     ; Get number of bit to check.
                   0005   30  2A3C  7995          BSBW    DISPLAY_QF              ; Convert to formatted string.
                01C0 8F   BA  2A3F  7996          POPR    #^M<R6,R7,R8>
                          05  2A43  7997          RSB
                              2A44  7998
                              2A44  7999 ;
                              2A44  8000 ; Convert the bit into an ascii string "YES" or "NO".
                              2A44  8001 ;
                              2A44  8002
                              2A44  8003 DISPLAY_QF:
                   02F6   30  2A44  8004          BSBW    CHECK_DIF               ; Check for change in data since last
                              2A47  8005                                          ;  update.
             04 67   56   E1  2A47  8006          BBC     R6,(R7),10$             ; Branch if QF_VOTE isn't set.
                     57   D4  2A4B  8007          CLRL    R7                      ; Get index into string array.
                     03   11  2A4D  8008          BRB     20$                     ; Branch to common.
                   57   01   D0  2A4F  8009 10$:   MOVL    #1,R7                   ; Get index into string array.
          58   00000404'EF   DE  2A52  8010 20$:   MOVAL   YES_NO_STRING,R8       ; Get address of string array.
                   FF8A   30  2A59  8011          BSBW    DISPLAY_STRING          ; Format the string.
                          05  2A5C  8012          RSB
                              2A5D  8013
```

```
                        2A5D  8015          .SBTTL  -       Systems class display routines.
                        2A5D  8016
                        2A5D  8017 ;++
                        2A5D  8018 ;
                        2A5D  8019 ; FUNCTIONAL DESCRIPTION:
                        2A5D  8020 ;
                        2A5D  8021 ;       This routine is used to combine SB$T_SWTYPE and SB$T_SWVERS to
                        2A5D  8022 ;       produce the SOFTWARE field.
                        2A5D  8023 ;
                        2A5D  8024 ; CALLING SEQUENCE:
                        2A5D  8025 ;
                        2A5D  8026 ;       JSB     DISPLAY_SOFTWARE
                        2A5D  8027 ;
                        2A5D  8028 ; CALLED BY:
                        2A5D  8029 ;
                        2A5D  8030 ;       GET_DATA
                        2A5D  8031 ;
                        2A5D  8032 ; INPUTS:
                        2A5D  8033 ;
                        2A5D  8034 ;       R7      = Offset to SB$T_SWVERS in the system block.
                        2A5D  8035 ;       (AP)[SB_IN]     = Address of system block.
                        2A5D  8036 ;
                        2A5D  8037 ; IMPLICIT INPUTS:
                        2A5D  8038 ;
                        2A5D  8039 ;       AP      = Address of call frame containing the addresses of the data
                        2A5D  8040 ;                 structures to be used when data is extracted.
                        2A5D  8041 ;
                        2A5D  8042 ; OUTPUTS:
                        2A5D  8043 ;
                        2A5D  8044 ;       Formated data string returned in FORMAT_BUF.
                        2A5D  8045 ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                        2A5D  8046 ;
                        2A5D  8047 ; RETURN STATUS:
                        2A5D  8048 ;
                        2A5D  8049 ;       Status returned from FORMAT_DATA routine.
                        2A5D  8050 ;
                        2A5D  8051 ; REGISTER USAGE:
                        2A5D  8052 ;
                        2A5D  8053 ;       R7      = Offset into data structure where data is located.  Converted
                        2A5D  8054 ;                 into actual address of data.
                        2A5D  8055 ;       R8      = Index into call frame to locate system block base address.
                        2A5D  8056 ;--
                        2A5D  8057
                        2A5D  8058 DISPLAY_SOFTWARE:
          0300 8F   BB  2A5D  8059          PUSHR   #^M<R8,R9>
             FF25   30  2A61  8060          BSBW    DISPLAY_FIELD           ; Format the software version.
59   0000215B'EF   D0  2A64  8061          MOVL    FORMAT_BUF,R9           ; Save the version temporarily.
          58   02   D0  2A6B  8062          MOVL    #SB_IN,R8               ; Set index into call frame to system
                    2A6E  8063                                              ;  block base address.
          57   24   D0  2A6E  8064          MOVL    #SB$T_SWTYPE,R7         ; Get software type offset.
        57   6C48   C0  2A71  8065          ADDL    (R12)[R8],R7            ; Get address of data.
0000215B'EF   67   D0  2A75  8066          MOVL    (R7),FORMAT_BUF         ; Store software type in buffer.
00C0215E'EF   20   90  2A7C  8067          MOVB    #^A/ /,FORMAT_BUF+3     ; Place a space between type and
                    2A83  8068                                              ;  version.
0000215F'EF   59   D0  2A83  8069          MOVL    R9,FORMAT_BUF+4         ; Store the version in the buffer.
00002153'EF   08   D0  2A8A  8070          MOVL    #8,FORMAT_BUF_LEN       ; Buffer length = 8 characters.
          0300 8F   BA  2A91  8071          POPR    #^M<R8,R9>
```

```
 05  2A95  8072          RSB
     2A96  8073
     2A96  8074
```

```
                        2A96  8076          .SBTTL  -       Cluster system block display routines.
                        2A96  8077
                        2A96  8078 ;++
                        2A96  8079 ;
                        2A96  8080 ; FUNCTIONAL DESCRIPTION:
                        2A96  8081 ;
                        2A96  8082 ;       This routine is used to display the protocol version and ECO level.
                        2A96  8083 ;
                        2A96  8084 ; CALLING SEQUENCE:
                        2A96  8085 ;
                        2A96  8086 ;       JSB     DISPLAY_PROTOCOL
                        2A96  8087 ;
                        2A96  8088 ; CALLED BY:
                        2A96  8089 ;
                        2A96  8090 ;       GET_DATA
                        2A96  8091 ;
                        2A96  8092 ; INPUTS:
                        2A96  8093 ;
                        2A96  8094 ;       R7      = SCS offset CSB$B_ECOLVL from DEFINE_FIELD macro.
                        2A96  8095 ;       (AP)[CSB_IN]    = Base address of selected cluster system block.
                        2A96  8096 ;
                        2A96  8097 ; IMPLICIT INPUTS:
                        2A96  8098 ;
                        2A96  8099 ;       AP      = Address of call frame containing the addresses of the data
                        2A96  8100 ;                 structures to be used when data is extracted.
                        2A96  8101 ;
                        2A96  8102 ; OUTPUTS:
                        2A96  8103 ;
                        2A96  8104 ;       Formated ascii string returned in FORMAT_BUF.
                        2A96  8105 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                        2A96  8106 ;
                        2A96  8107 ; RETURN STATUS:
                        2A96  8108 ;
                        2A96  8109 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                        2A96  8110 ;
                        2A96  8111 ;--
                        2A96  8112
                        2A96  8113 DISPLAY_PROTOCOL:
              0180 8F   BB  2A96  8114          PUSHR   #^M<R7,R8>
              58   03   D0  2A9A  8115          MOVL    #CSB_IN,R8              ; Set index into call frame to path
                        2A9D  8116                                              ;  block base address.
          57  6C48   C0  2A9D  8117          ADDL    (R12)[R8],R7            ; Get the address of the data.
                0299  30  2AA1  8118          BSBW    CHECK_DIF               ; Check for change in data since last
                        2AA4  8119                                              ;  scan.
00000018'EF   67   08   78  2AA4  8120          ASHL    #8,(R7),SCRATCH         ; Store eco level in 2nd byte of
                        2AAC  8121                                              ;  scratch location.
    57  00000041 8F   D0  2AAC  8122          MOVL    #CSB$B_VERNUM,R7        ; Get version number offset.
          57  6C48   C0  2AB3  8123          ADDL    (R12)[R8],R7            ; Get the address of the data.
                0283  30  2AB7  8124          BSBW    CHECK_DIF               ; Check for change in data since last
                        2ABA  8125                                              ;  scan.
00000018'EF   67   90   2ABA  8126          MOVB    (R7),SCRATCH            ; Store version number in 1st byte of
                        2AC1  8127                                              ;  scratch location.
    57  00000018'EF   DE  2AC1  8128          MOVAL   SCRATCH,R7              ; Get address of data.
                FAB4  30  2AC8  8129          BSBW    FORMAT_DATA             ; Format the data.
              0180 8F   BA  2ACB  8130          POPR    #^M<R7,R8>
                        05  2ACF  8131          RSB
                        2AD0  8132
```

2ADO 8133

```
                     2AD0  8135
                     2AD0  8136 ;++
                     2AD0  8137 ;
                     2AD0  8138 ; FUNCTIONAL DESCRIPTION:
                     2AD0  8139 ;
                     2AD0  8140 ;       This routine is used to display the state of the cluster system.
                     2AD0  8141 ;
                     2AD0  8142 ; CALLING SEQUENCE:
                     2AD0  8143 ;
                     2AD0  8144 ;       JSB     DISPLAY_CSB_STATE
                     2AD0  8145 ;
                     2AD0  8146 ; CALLED BY:
                     2AD0  8147 ;
                     2AD0  8148 ;       GET_DATA
                     2AD0  8149 ;
                     2AD0  8150 ; INPUTS:
                     2AD0  8151 ;
                     2AD0  8152 ;       R7      = SCS offset CSB$B_STATE from DEFINE_FIELD macro.
                     2AD0  8153 ;       (AP)[CSB_IN]    = Base address of selected cluster system block.
                     2AD0  8154 ;
                     2AD0  8155 ; IMPLICIT INPUTS:
                     2AD0  8156 ;
                     2AD0  8157 ;       AP      = Address of call frame containing the addresses of the data
                     2AD0  8158 ;                 structures to be used when data is extracted.
                     2AD0  8159 ;
                     2AD0  8160 ; OUTPUTS:
                     2AD0  8161 ;
                     2AD0  8162 ;       Formated ascii string returned in FORMAT_BUF.
                     2AD0  8163 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2AD0  8164 ;
                     2AD0  8165 ; RETURN STATUS:
                     2AD0  8166 ;
                     2AD0  8167 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2AD0  8168 ;
                     2AD0  8169 ;--
                     2AD0  8170
                     2AD0  8171 DISPLAY_CSB_STATE:
          0180 8F    BB  2AD0  8172          PUSHR   #^M<R7,R8>
          58   03    D0  2AD4  8173          MOVL    #CSB_IN,R8              ; Set index into call frame to path
                           2AD7  8174                                          ;  block base address.
      57   6C48    C0  2AD7  8175          ADDL    (R12)[R8],R7            ; Get the address of the data.
           025F    30  2ADB  8176          BSBW    CHECK_DIF               ; Check for change in data since last
                           2ADE  8177                                          ;  scan.
58  0000040B'EF    DE  2ADE  8178          MOVAL   CSB_STATE_STRNG,R8      ; Point to array of state descriptors.
      57   67    9A  2AE5  8179          MOVZBL  (R7),R7                 ; Get the state.
           57    D7  2AE8  8180          DECL    R7                      ; Subtract one so index starts at zero.
         FEF9    30  2AEA  8181          BSBW    DISPLAY_STRING          ; Format the string.
          0180 8F    BA  2AED  8182          POPR    #^M<R7,R8>
                     05  2AF1  8183          RSB
                         2AF2  8184
                         2AF2  8185
```

```
                        2AF2  8187
                        2AF2  8188 ;++
                        2AF2  8189 ;
                        2AF2  8190 ; FUNCTIONAL DESCRIPTION:
                        2AF2  8191 ;
                        2AF2  8192 ;       This routine is used to display the state of the cluster system.
                        2AF2  8193 ;
                        2AF2  8194 ;
                        2AF2  8195 ; CALLING SEQUENCE:
                        2AF2  8196 ;
                        2AF2  8197 ;       JSB     DISPLAY_CSB_STATUS
                        2AF2  8198 ;
                        2AF2  8199 ; CALLED BY:
                        2AF2  8200 ;
                        2AF2  8201 ;       GET_DATA
                        2AF2  8202 ;
                        2AF2  8203 ; INPUTS:
                        2AF2  8204 ;
                        2AF2  8205 ;       R7      = SCS offset CSB$L_STATUS from DEFINE_FIELD macro.
                        2AF2  8206 ;       (AP)[CSB_IN]    = Base address of selected cluster system block.
                        2AF2  8207 ;
                        2AF2  8208 ; IMPLICIT INPUTS:
                        2AF2  8209 ;
                        2AF2  8210 ;       AP      = Address of call frame containing the addresses of the data
                        2AF2  8211 ;                 structures to be used when data is extracted.
                        2AF2  8212 ;
                        2AF2  8213 ; OUTPUTS:
                        2AF2  8214 ;
                        2AF2  8215 ;       Formated ascii string returned in FORMAT_BUF.
                        2AF2  8216 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                        2AF2  8217 ;
                        2AF2  8218 ; RETURN STATUS:
                        2AF2  8219 ;
                        2AF2  8220 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                        2AF2  8221 ;
                        2AF2  8222 ;--
                        2AF2  8223
                        2AF2  8224 ASSUME CSB$V_LONG_BREAK EQ 0
                        2AF2  8225 ASSUME CSB$V_MEMBER     EQ 1
                        2AF2  8226 ASSUME CSB$V_REMOVED    EQ 2
                        2AF2  8227
                        2AF2  8228 DISPLAY_CSB_STATUS:
          0180 8F   BB  2AF2  8229         PUSHR   #^M<R7,R8>
          58   03   D0  2AF6  8230         MOVL    #CSB_IN,R8              ; Set index into call frame to path
                        2AF9  8231                                         ;  block base address.
       57   6C48   C0  2AF9  8232         ADDL    (R12)[R8],R7            ; Get the address of the data.
            023D   30  2AFD  8233         BSBW    CHECK_DIF               ; Check for change in data since last
                        2B00  8234                                         ;  scan.
       57   67     90  2B00  8235         MOVB    (R7),R7                 ; Get the status.
57  FFFFFFF8 8F   CA  2B03  8236         BICL    #^C<CSB$M_LONG_BREAK!-  ; Clear any non-status bits.
                        2B0A  8237                 CSB$M_MEMBER!-          ;
                        2B0A  8238                 CSB$M_REMOVED>,R7       ;
58  00000452'FF   DE  2B0A  8239         MOVAL   CSB_STATUS_STRNG,R8     ; Point to array of status descriptors.
            FED2   30  2B11  8240         BSBW    DISPLAY_STRING          ; Format the string.
          0180 8F   BA  2B14  8241         POPR    #^M<R7,R8>
                   05  2B18  8242         RSB
                        2B19  8243
```

```
                        2819  8245          .SBTTL  -       Circuits class display routines.
                        2819  8246
                        2819  8247 ;++
                        2819  8248 ;
                        2819  8249 ; FUNCTIONAL DESCRIPTION:
                        2819  8250 ;
                        2819  8251 ;       This routine is used to display the remote port type using the data
                        2819  8252 ;       stored at offset PB$L_RPORT_TYP in the path block.  CI780s and CI750s
                        2819  8253 ;       both have a port type of 2.  If PB$L_RPORT_TYP indicates a port type
                        2819  8254 ;       code = 2, then the hardware type field in the system block for the
                        2819  8255 ;       remote system is checked to make the distinction.
                        2819  8256 ;
                        2819  8257 ;
                        2819  8258 ; CALLING SEQUENCE:
                        2819  8259 ;
                        2819  8260 ;       JSB     DISPLAY_RPORT_TYP
                        2819  8261 ;
                        2819  8262 ; CALLED BY:
                        2819  8263 ;
                        2819  8264 ;       GET_DATA
                        2819  8265 ;
                        2819  8266 ; INPUTS:
                        2819  8267 ;
                        2819  8268 ;       R7  =   SCS offset PB$L_RPORT_TYP from DEFINE FIELD macro.
                        2819  8269 ;       (AP)[PB_IN]     = Address of path block.
                        2819  8270 ;       (AP)[SB_IN]     = Address of system block.
                        2819  8271 ;
                        2819  8272 ; IMPLICIT INPUTS:
                        2819  8273 ;
                        2819  8274 ;       AP      = Address of call frame containing the addresses of the data
                        2819  8275 ;                 structures to be used when data is extracted.
                        2819  8276 ;
                        2819  8277 ; OUTPUTS:
                        2819  8278 ;
                        2819  8279 ;       Formated ascii string returned in FORMAT_BUF.
                        2819  8280 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                        2819  8281 ;
                        2819  8282 ; RETURN STATUS:
                        2819  8283 ;
                        2819  8284 ;       R0  =   SS$_NORMAL of failed status returned from FORMAT_DATA.
                        2819  8285 ;
                        2819  8286 ;--
                        2819  8287
                        2819  8288 DISPLAY_RPORT_TYP:
      0180 8F   BB      2819  8289          PUSHR   #^M<R7,R8>
      58   04   D0      2B1D  8290          MOVL    #PB_IN,R8               ; Set index into call frame to path
                        2B20  8291                                          ;  block base address.
   57   6C48   C0      2B20  8292          ADDL    (R12)[R8],R7            ; Get the address of the data.
        0216   30      2B24  8293          BSBW    CHECK_DIF               ; Check for change in data since last
                        2B27  8294                                          ;  scan.
          00   EF      2B27  8295          EXTZV   #PB$V_PORT_TYP,-        ; Get the port type code.
57   67   1F            2B29  8296                  #PB$S_PORT_TYP,(R7),R7  ;
      57   02   C2      2B2C  8297          SUBL    #2,R7                   ; Convert port type code to index into
                        2B2F  8298                                          ;  array of port type descriptors.
          28   12      2B2F  8299          BNEQ    10$                     ; Branch if port is not a CI780 or
                        2B31  8300                                          ;  CI750.
      58   02   D0      2B31  8301          MOVL    #SB_IN,R8               ; Set index into call frame to system
```

```
                        2B34  8302
                              8303                              ;  block base address.
        58    6C48  D0  2B34        MOVL    (R12)[R8],R8        ; Get the address of the system block
                        2B38  8304                              ;  for the remote system.
        58    34 A8   D0  2B38  8305        MOVL    SB$T_HWTYPE(R8),R8  ; Get the remote system hardware type.
30383756 8F    58     D1  2B3C  8306        CMPL    R8,#^A/V780/        ; Is the remote system an 11/780?
                14     13  2B43  8307        BEQL    10$                 ; Branch if so.
35383756 8F    58     D1  2B45  8308        CMPL    R8,#^A/V785/        ; Is the remote system an 11/785?
                0B     13  2B4C  8309        BEQL    10$                 ; Branch if so.
                57     D6  2B4E  8310        INCL    R7                  ; Bump port type index.
30353756 8F    58     D1  2B50  8311        CMPL    R8,#^A/V750/        ; Is the remote system an 11/750?
                0A     12  2B57  8312        BNEQ    99$                 ; Error if not.
58   000002CA'EF   DE  2B59  8313 10$:     MOVAL   RPORT_TYP_STRNG,R8  ; Point to array of port type
                        2B60  8314                              ;  descriptors.
             FE83   30  2B60  8315        BSBW    DISPLAY_STRING      ; Display the port type string.
          0180 8F   BA  2B63  8316 99$:    POPR    #^M<R7,R8>
                    05  2B67  8317        RSB
                        2B68  8318
                        2B68  8319
```

```
                     2B68  8321 ;++
                     2B68  8322 ;
                     2B68  8323 ; FUNCTIONAL DESCRIPTION:
                     2B68  8324 ;
                     2B68  8325 ;       This routine is used to display the state of the virtual circuit.
                     2B68  8326 ;
                     2B68  8327 ; CALLING SEQUENCE:
                     2B68  8328 ;
                     2B68  8329 ;       JSB     DISPLAY_CIR_STATE
                     2B68  8330 ;
                     2B68  8331 ; CALLED BY:
                     2B68  8332 ;
                     2B68  8333 ;       GET_DATA
                     2B68  8334 ;
                     2B68  8335 ; INPUTS:
                     2B68  8336 ;
                     2B68  8337 ;       R7      = SCS Offset PB$W_STATE from DEFINE_FIELD macro.
                     2B68  8338 ;       (AP)[PB_IN]     = Base address of selected path block.
                     2B68  8339 ;
                     2B68  8340 ; IMPLICIT INPUTS:
                     2B68  8341 ;
                     2B68  8342 ;       AP      = Address of call frame containing the addresses of the data
                     2B68  8343 ;                 structures to be used when data is extracted.
                     2B68  8344 ;
                     2B68  8345 ; OUTPUTS:
                     2B68  8346 ;
                     2B68  8347 ;       Formated ascii string returned in FORMAT_BUF.
                     2B68  8348 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2B68  8349 ;
                     2B68  8350 ; RETURN STATUS:
                     2B68  8351 ;
                     2B68  8352 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2B68  8353 ;
                     2B68  8354 ;--
                     2B68  8355
                     2B68  8356 DISPLAY_CIR_STATE:
          0180 8F    BB  2B68  8357          PUSHR   #^M<R7,R8>
          58   04    D0  2B6C  8358          MOVL    #PB_IN,R8               ; Set index into call frame to path
                     2B6F  8359                                              ;  block base address.
       57   6C48    C0  2B6F  8360          ADDL    (R12)[R8],R7            ; Get address of data.
            01C7    30  2B73  8361          BSBW    CHECK_DIF               ; Check for change in data since last
                     2B76  8362                                              ;  scan.
58  0000030D'EF    DE  2B76  8363          MOVAL   PB_STATE_STRNG,R8       ; Point to array of circuit state
                     2B7D  8364                                              ;  strings.
          57   67    3C  2B7D  8365          MOVZWL  (R7),R7                 ; Get the state.
       57   8000 8F    B3  2B80  8366          BITW    #PB$C_VC_FAIL,R7        ; Check for VC_FAIL.
            03    13  2B85  8367          BEQLU   10$                     ; Branch if not.
          57   04    B0  2B87  8368          MOVW    #4,R7                   ; Get index to VC_FAIL string.
             FE59    30  2B8A  8369 10$:     BSBW    DISPLAY_STRING          ; Display the string.
          0180 8F    BA  2B8D  8370          POPR    #^M<R7,R8>
                05    2B91  8371          RSB
                     2B92  8372
                     2B92  8373
```

```
                     2B92  8375 ;++
                     2B92  8376 ;
                     2B92  8377 ; FUNCTIONAL DESCRIPTION:
                     2B92  8378 ;
                     2B92  8379 ;       This routine is used to display the state of the remote port.
                     2B92  8380 ;
                     2B92  8381 ; CALLING SEQUENCE:
                     2B92  8382 ;
                     2B92  8383 ;       JSB     DISPLAY_RSTATE
                     2B92  8384 ;
                     2B92  8385 ; CALLED BY:
                     2B92  8386 ;
                     2B92  8387 ;       GET_DATA
                     2B92  8388 ;
                     2B92  8389 ; INPUTS:
                     2B92  8390 ;
                     2B92  8391 ;       R7      = SCS offset PB$B_RSTATE from DEFINE_FIELD macro.
                     2B92  8392 ;       (AP)[PB_IN]     = Base address of selected path block.
                     2B92  8393 ;
                     2B92  8394 ; IMPLICIT INPUTS:
                     2B92  8395 ;
                     2B92  8396 ;       AP      = Address of call frame containing the addresses of the data
                     2B92  8397 ;                 structures to be used when data is extracted.
                     2B92  8398 ;
                     2B92  8399 ; OUTPUTS:
                     2B92  8400 ;
                     2B92  8401 ;       Formated ascii string returned in FORMAT_BUF.
                     2B92  8402 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2B92  8403 ;
                     2B92  8404 ; RETURN STATUS:
                     2B92  8405 ;
                     2B92  8406 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2B92  8407 ;
                     2B92  8408 ;--
                     2B92  8409
                     2B92  8410 DISPLAY_RSTATE:
          0180 8F    BB  2B92  8411          PUSHR   #^M<R7,R8>
          58   04    D0  2B96  8412          MOVL    #PB_IN,R8               ; Set index into call frame to path
                     2B99  8413                                              ;  block base address.
      57   6C48    C0  2B99  8414          ADDL    (R12)[R8],R7            ; Get the address of the data.
           019D    30  2B9D  8415          BSBW    CHECK_DIF               ; Check for change in data since last
                     2BA0  8416                                              ;  scan.
58 00000330'EF    DE  2BA0  8417          MOVAL   PB_RSTATE_STRNG,R8      ; Point to array of state descriptors.
      57   67    9A  2BA7  8418          MOVZBL  (R7),R7                 ; Get the state.
         FE39    30  2BAA  8419          BSBW    DISPLAY_STRING          ; Format the string.
          0180 8F    BA  2BAD  8420          POPR    #^M<R7,R8>
                     05  2BB1  842i          RSB
                     2BB2  8422
                     2BB2  8423
```

```
                     ;++
                     ;
                     ; FUNCTIONAL DESCRIPTION:
                     ;
                     ;       This routine is used to display the path/cable status.  The state of
                     ;       the cable is first checked and if good the states of the path A and path
                     ;       B are checked.  The output is in the format:
                     ;
                     ;               "  -  "         Cable is good.  No good paths.
                     ;               "A -  "         Cable is good.  Path A is good.
                     ;               "  - B"         Cable is good.  Path B is good.
                     ;               "A - B"         Cable is good.  Paths A and B are gocd.
                     ;               "CROSSED"       Cable is bad or crossed.
                     ;
                     ; CALLING SEQUENCE:
                     ;
                     ;       JSB     DISPLAY_CABLE_STATUS
                     ;
                     ; CALLED BY:
                     ;
                     ;       GET_DATA
                     ;
                     ; INPUTS:
                     ;
                     ;       R7      = SCS offset PB$B_CBL_STS from DEFINE_FIELD macro.
                     ;       (AP)[PB_IN]     = Base address of selected path block.
                     ;
                     ; IMPLICIT INPUTS:
                     ;
                     ;       AP      = Address of call frame containing the addresses of the data
                     ;                 structures to be used when data is extracted.
                     ;
                     ; OUTPUTS:
                     ;
                     ;       Formated ascii string returned in FORMAT_BUF.
                     ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     ;
                     ; RETURN STATUS:
                     ;
                     ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     ;
                     ;--

                     ASSUME PB$B_P0_STS      EQ PB$B_CBL_STS+1
                     ASSUME PB$B_P1_STS      EQ PB$B_P0_STS+1

                     DISPLAY_CABLE_STATUS:
 0380 8F     BB      PUSHR   #^M<R7,R8,R9>
 58   04     D0      MOVL    #PB_IN,R8               ; Set index into call frame to path
                                                     ;  block base address.
 57   6C48   C0      ADDL    (R12)[R8],R7            ; Get the address of the data.
      017D   30      BSBW    CHECK_DIF               ; Check for change in data since last
                                                     ;  scan.
58 0000028C'EF      MOVAL   CB_STAT_STRG,R8         ; Point to cable status strings.
 87   01     93      BITB    #PB$M_COR_CBL,(R7)+     ; Is the cable status ok?
      05     12      BNEQ    10$                     ; Branch if so.
 57   04     D0      MOVL    #4,R7                   ; Indicate crossed.
```

```
                1B   11  2BCF  8482          BRB     40$                     ; Skip path state checks.
            0169     30  2BD1  8483 10$:     BSBW    CHECK_DIF               ; Check for change in path A.
              59     D4  2BD4  8484          CLRL    R9                      ; Assume path A and path B bad.
87            01     93  2BD6  8485          BITB    #PB$M_CUR_PS,(R7)+      ; Path A ok?
              03     13  2BD9  8486          BEQL    20$                     ; Branch if not.
59            01     C8  2BDB  8487          BISL    #1,R9                   ; Indicate path A ok.
            015C     30  2BDE  8488 20$:     BSBW    CHECK_DIF               ; Check for change in path B.
67            01     93  2BE1  8489          BITB    #PB$M_CUR_PS,(R7)       ; Path B ok?
              03     13  2BE4  8490          BEQL    30$                     ; Branch if not.
59            02     C8  2BE6  8491          BISL    #2,R9                   ; Indicate B ok.
57            59     D0  2BE9  8492 30$:     MOVL    R9,R7                   ; Get result in R7.
            FDF7     30  2BEC  8493 40$:     BSBW    DISPLAY_STRING          ; Display the string.
0380        8F     BA  2BEF  8494          POPR    #^M<R7,R8,R9>
                   05  2BF3  8495          RSB
                       2BF4  8496
                       2BF4  8497
```

```
                     2BF4  8499 ;++
                     2BF4  8500 ;
                     2BF4  8501 ; FUNCTIONAL DESCRIPTION:
                     2BF4  8502 ;
                     2BF4  8503 ;       This routine is used to display the local or remote process names
                     2BF4  8504 ;       associated with a CDT.  The process names, if available, are stored
                     2BF4  8505 ;       following each CDT by the snapshot routine.  The first 16 bytes being
                     2BF4  8506 ;       the local process name and the next 16 bytes being the remote process
                     2BF4  8507 ;       name.  The routine determines which name is to be formatted by the
                     2BF4  8508 ;       offset in R7.
                     2BF4  8509 ;
                     2BF4  8510 ; CALLING SEQUENCE:
                     2BF4  8511 ;
                     2BF4  8512 ;       JSB     DISPLAY_PROC_NAME
                     2BF4  8513 ;
                     2BF4  8514 ; CALLED BY:
                     2BF4  8515 ;
                     2BF4  8516 ;       GET_DATA
                     2BF4  8517 ;
                     2BF4  8518 ; INPUTS:
                     2BF4  8519 ;
                     2BF4  8520 ;       R7      = SCS offset SCS$T_DST_PROC for local process name or
                     2BF4  8521 ;                 SCS$T_SRC_PROC for remote process name.  From DEFINE_FIELD
                     2BF4  8522 ;                 macro.
                     2BF4  8523 ;       (AP)[CDT_IN]    = Base address of selected CDT.
                     2BF4  8524 ;
                     2BF4  8525 ; IMPLICIT INPUTS:
                     2BF4  8526 ;
                     2BF4  8527 ;       AP      = Address of call frame containing the addresses of the data
                     2BF4  8528 ;                 structures to be used when data is extracted.
                     2BF4  8529 ;
                     2BF4  8530 ; OUTPUTS:
                     2BF4  8531 ;
                     2BF4  8532 ;       Formated ascii string returned in FORMAT_BUF.
                     2BF4  8533 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2BF4  8534 ;
                     2BF4  8535 ; RETURN STATUS:
                     2BF4  8536 ;
                     2BF4  8537 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2BF4  8538 ;
                     2BF4  8539 ;--
                     2BF4  8540
                     2BF4  8541 ASSUME SCS$T_SRC_PROC   EQ SCS$T_DST_PROC+16
                     2BF4  8542
                     2BF4  8543 DISPLAY_PROC_NAME:
          0180 8F    BB  2BF4  8544          PUSHR   #^M<R7,R8>
               50    D4  2BF8  8545          CLRL    R0                      ; Assume failure.
               58    05  D0  2BFA  8546          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                         2BFD  8547                                          ;  block base address.
58  0000008C 8F    6C48  C1  2BFD  8548          ADDL3   (R12)[R8],-             ; Get the address of the process names
                         2C06  8549                  #SHC$C_CDT_LENGTH,R8    ;  associated with the CDT by adding
                         2C06  8550                                          ;  the CDT base address to the length
                         2C06  8551                                          ;  of the CDT.
               57    04  D1  2C06  8552          CMPL    #SCS$T_DST_PROC,R7      ; Are we displaying the local process
                         2C09  8553                                          ;  name?
                     08  13  2C09  8554          BEQL    10$                     ; Branch if so.
               57    14  D1  2C0B  8555          CMPL    #SCS$T_SRC_PROC,R7      ; If not, make sure we are displaying
```

```
                     2C0E  8556                                         ;  the remote process name.
          0C   12    2C0E  8557          BNEQ    99$                    ; Error if not.
       58 10   C0    2C10  8558          ADDL    #16,R8                 ; Point to remote process name.
       57 58   D0    2C13  8559 10$:     MOVL    R8,R7                  ; Get address of data in R7.
        0124   30    2C16  8560          BSBW    CHECK_DIF              ; Check for change in data since last
                     2C19  8561                                         ;  scan.
        F963   30    2C19  8562          BSBW    FORMAT_DATA            ; Format the string.
     0180 8F   BA    2C1C  8563 99$:     POPR    #^M<R7,R8>
               05    2C20  8564          RSB
                     2C21  8565
                     2C21  8566
                     2C21  8567 ;++
                     2C21  8568 ;
                     2C21  8569 ; FUNCTIONAL DESCRIPTION:
                     2C21  8570 ;
                     2C21  8571 ;       This routine is used to display the state of a connection.
                     2C21  8572 ;
                     2C21  8573 ; CALLING SEQUENCE:
                     2C21  8574 ;
                     2C21  8575 ;       JSB     DISPLAY_CON_STATE
                     2C21  8576 ;
                     2C21  8577 ; CALLED BY:
                     2C21  8578 ;
                     2C21  8579 ;       GET_DATA
                     2C21  8580 ;
                     2C21  8581 ; INPUTS:
                     2C21  8582 ;
                     2C21  8583 ;       R7      = SCS offset CDT$W_STATE from DEFINE_FIELD macro.
                     2C21  8584 ;       (AP)[CDT_IN]    = Base address of selected CDT.
                     2C21  8585 ;
                     2C21  8586 ; IMPLICIT INPUTS:
                     2C21  8587 ;
                     2C21  8588 ;       AP      = Address of call frame containing the addresses of the data
                     2C21  8589 ;                 structures to be used when data is extracted.
                     2C21  8590 ;
                     2C21  8591 ; OUTPUTS:
                     2C21  8592 ;
                     2C21  8593 ;       Formated ascii string returned in FORMAT_BUF.
                     2C21  8594 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2C21  8595 ;
                     2C21  8596 ; RETURN STATUS:
                     2C21  8597 ;
                     2C21  8598 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2C21  8599 ;
                     2C21  8600 ;--
                     2C21  8601
                     2C21  8602 DISPLAY_CON_STATE:
        0180 8F   BB 2C21  8603          PUSHR   #^M<R7,R8>
        58   05   D0 2C25  8604          MOVL    #CDT_IN,R8                 ; Set index into call frame to CDT
                     2C28  8605                                             ;  block base address.
57 FFFFFFF0 8F   C0 2C28  8606          ADDL    #SHC$L_CDT_DIF,R7          ; Adjust CDT offset to compensate for
                     2C2F  8607                                             ;  portion of CDT not stored.
     57 6C48     C0 2C2F  8608          ADDL    (R12)[R8],R7               ; Get the address of the data.
           0107  30 2C33  8609          BSBW    CHECK_DIF                  ; Check for change in data since last
                     2C36  8610                                             ;  scan.
58 0000035A'EF   DE 2C36  8611          MOVAL   CDT_STATE_STRNG,R8         ; Point to array of connection state
                     2C3D  8612                                             ;  strings.
```

```
                57   67   3C  2C3D  8613          MOVZWL  (R7),R7                  ; Get the connection state.
                   FDA3   30  2C40  8614          BSBW    DISPLAY_STRING          ; Display the string.
                0180 8F   BA  2C43  8615          POPR    #^M<R7,R8>
                          05  2C47  8616          RSB
                              2C48  8617
                              2C48  8618
```

```
                     2C48  8620 ;++
                     2C48  8621 ;
                     2C48  8622 ; FUNCTIONAL DESCRIPTION:
                     2C48  8623 ;
                     2C48  8624 ;       This routine is used to display the SCS blocked state.
                     2C48  8625 ;
                     2C48  8626 ; CALLING SEQUENCE:
                     2C48  8627 ;
                     2C48  8628 ;       JSB     DISPLAY_SCS_STATE
                     2C48  8629 ;
                     2C48  8630 ; CALLED BY:
                     2C48  8631 ;
                     2C48  8632 ;       GET_DATA
                     2C48  8633 ;
                     2C48  8634 ; INPUTS:
                     2C48  8635 ;
                     2C48  8636 ;       R7      = SCS offset CDT$W_BLKSTATE from DEFINE_FIELD macro.
                     2C48  8637 ;       (AP)[CDT_IN]    = Base address of selected CDT.
                     2C48  8638 ;
                     2C48  8639 ; IMPLICIT INPUTS:
                     2C48  8640 ;
                     2C48  8641 ;       AP      = Address of call frame containing the addresses of the data
                     2C48  8642 ;                 structures to be used when data is extracted.
                     2C48  8643 ;
                     2C48  8644 ; OUTPUTS:
                     2C48  8645 ;
                     2C48  8646 ;       Formated ascii string returned in FORMAT_BUF.
                     2C48  8647 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                     2C48  8648 ;
                     2C48  8649 ; RETURN STATUS:
                     2C48  8650 ;
                     2C48  8651 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                     2C48  8652 ;
                     2C48  8653 ;--
                     2C48  8654
                     2C48  8655 DISPLAY_SCS_STATE:
          0180 8F    BB  2C48  8656          PUSHR   #^M<R7,R8>
          58   05    D0  2C4C  8657          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                     2C4F  8658                                              ;  block base address.
57   FFFFFFF0 8F    C0  2C4F  8659          ADDL    #SMC$L_CDT_DIF,R7       ; Adjust CDT offset to compensate for
                     2C56  8660                                              ;  portion of CDT not stored.
       57   6C48    C0  2C56  8661          ADDL    (R12)[R8],R7            ; Get the address of the data.
            00E0    30  2C5A  8662          BSBW    CHECK_DIF               ; Check for change in data since last
                     2C5D  8663                                              ;  scan.
58   00000307'EF    DE  2C5D  8664          MOVAL   CDT_BLKSTATE_STRNG,R8   ; Point to array of send blocked state
                     2C64  8665                                              ;  strings.
       57   67      3C  2C64  8666          MOVZWL  (R7),R7                 ; Get the SCS blocked state.
              FD7C    30  2C67  8667          BSBW    DISPLAY_STRING          ; Display the string.
          0180 8F    BA  2C6A  8668          POPR    #^M<R7,R8>
                     05  2C6E  8669          RSB
                     2C6F  8670
                     2C6F  8671
                     2C6F  8672          .SBTTL  -       Local port display routines.
```

```
                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine is used to display the kilobyte values CDT$L_KBYTSENT,
                                ;       CDT$L_KBYTREQD, and CDT$L_KBYTMAPD.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     DISPLAY_KBYTES
                                ;
                                ; CALLED BY:
                                ;
                                ;       GET_DATA
                                ;
                                ; INPUTS:
                                ;
                                ;       R7      = SCS offset from DEFINE_FIELD macro.
                                ;       (AP)[CDT_IN]    = Base address of selected CDT.
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       AP      = Address of call frame containing the addresses of the data
                                ;                 structures to be used when data is extracted.
                                ;
                                ; OUTPUTS:
                                ;
                                ;       Formated ascii string returned in FORMAT_BUF.
                                ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                                ;
                                ; RETURN STATUS:
                                ;
                                ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                                ;
                                ;--
                                
                                DISPLAY_KBYTES:
                0180 8F   BB    PUSHR   #^M<R7,R8>
                58   05   D0    MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                                                                ;  block base address.
     57  FFFFFFF0 8F   C0    ADDL    #SHC$L_CDT_DIF,R7       ; Adjust CDT offset to compensate for
                                                                ;  portion of CDT not stored.
           57  6C48   C0    ADDL    (R12)[R8],R7            ; Get the address of the data.
67   67   16   0A   EF    EXTZV   #10,#22,(R7),(R7)       ; Convert bytes to Kbytes.
               00B4   30    BSBW    CHECK_DIF               ; Check for change in data since last
                                                                ;  scan.
                F8F3   30    BSBW    FORMAT_DATA             ; Format the string.
                0180 8F   BA    POPR    #^M<R7,R8>
                       05    RSB
                                
                                
```

```
                        2C91  8726 ;++
                        2C91  8727 ;
                        2C91  8728 ; FUNCTIONAL DESCRIPTION:
                        2C91  8729 ;
                        2C91  8730 ;       This routine is used to display the status of the port.  The value
                        2C91  8731 ;       at offset MB$L_STS for the selected port in the miscellaneous buffer
                        2C91  8732 ;       is checked for online or offline status.  A string is then placed in
                        2C91  8733 ;       the format buffer which is an ascii representation of the status.
                        2C91  8734 ;
                        2C91  8735 ; CALLING SEQUENCE:
                        2C91  8736 ;
                        2C91  8737 ;       JSB     DISPLAY_PORT_STATUS
                        2C91  8738 ;
                        2C91  8739 ; CALLED BY:
                        2C91  8740 ;
                        2C91  8741 ;       GET_DATA
                        2C91  8742 ;
                        2C91  8743 ; INPUTS:
                        2C91  8744 ;
                        2C91  8745 ;       R7      = Offset MB$L_STS from DEFINE_FIELD macro.
                        2C91  8746 ;       (AP)[MB_IN] = Base address of the data for selected port in MISC_BUF.
                        2C91  8747 ;
                        2C91  8748 ; IMPLICIT INPUTS:
                        2C91  8749 ;
                        2C91  8750 ;       AP      = Address of call frame containing the addresses of the data
                        2C91  8751 ;                 structures to be used when data is extracted.
                        2C91  8752 ;
                        2C91  8753 ; OUTPUTS:
                        2C91  8754 ;
                        2C91  8755 ;       Formated ascii string returned in FORMAT_BUF.
                        2C91  8756 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                        2C91  8757 ;
                        2C91  8758 ; RETURN STATUS:
                        2C91  8759 ;
                        2C91  8760 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                        2C91  8761 ;
                        2C91  8762 ;--
                        2C91  8763
                        2C91  8764 DISPLAY_PORT_STATUS:
          0180 8F   BB  2C91  8765          PUSHR   #^M<R7,R8>
          58   07   D0  2C95  8766          MOVL    #MB_IN,R8                       ; Set index into call frame to base
                        2C98  8767                                                  ;  address of port data in MISC_BUF.
       57   6C48   C0  2C98  8768          ADDL    (R12)[R8],R7                    ; Get address of data.
            009E   30  2C9C  8769          BSBW    CHECK_DIF                       ; Check for change in data since last
                        2C9F  8770                                                  ;  scan.
58  000002FE'EF   DE  2C9F  8771          MOVAL   PORT_STATUS_STRNG,R8            ; Get address of port status strings.
       03 67   04   E1  2CA6  8772          BBC     #UCB$V_ONLINE,(R7),10$          ; Branch if the port is offline.
          58   88   80  2CAA  8773          ADDB    (R8)+,R8                        ; Otherwise, point to online status
                        2CAD  8774                                                  ;  string.
          57   58   D0  2CAD  8775 10$:     MOVL    R8,R7                           ; Get address of string.
               F8CC   30  2CB0  8776          BSBW    FORMAT_DATA                     ; Format the string.
          0180 8F   BA  2CB3  8777          POPR    #^M<R7,R8>
                   05   2CB7  8778          RSB
                        2CB8  8779
                        2CB8  8780
```

```
                                                ;++
                                                ;
                                                ; FUNCTIONAL DESCRIPTION:
                                                ;
                                                ;       This routine is used to display the device type of the port.  The value
                                                ;       at offset MB$L_DEVTYPE is checked for device type of CI780 or CI750.
                                                ;       The appropriate string is then stored in the format buffer.
                                                ;
                                                ; CALLING SEQUENCE:
                                                ;
                                                ;       JSB     DISPLAY_PORT_TYPE
                                                ;
                                                ; CALLED BY:
                                                ;
                                                ;       GET_DATA
                                                ;
                                                ; INPUTS:
                                                ;
                                                ;       R7      = Offset MB$L_DEVTYPE from DEFINE_FIELD macro.
                                                ;       (AP)[MB_IN] = Base address of the data for selected port in MISC_BUF.
                                                ;
                                                ; IMPLICIT INPUTS:
                                                ;
                                                ;       AP      = Address of call frame containing the addresses of the data
                                                ;                 structures to be used when data is extracted.
                                                ;
                                                ; OUTPUTS:
                                                ;
                                                ;       Formated ascii string returned in FORMAT_BUF.
                                                ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                                                ;
                                                ; RETURN STATUS:
                                                ;
                                                ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                                                ;
                                                ;--
                                                
                                                DISPLAY_PORT_TYPE:
0180 8F     BB  PUSHR   #^M<R7,R8>
58   07     D0  MOVL    #MB_IN,R8                   ; Set index into call frame to base
                                                    ;  address of port data in MISC_BUF.
57   6C48   C0  ADDL    (R12)[R8],R7                ; Get address of data.
     0077   30  BSBW    CHECK_DIF                   ; Check for change in data since last
                                                    ;  scan.
58 000002F2'EF  DE  MOVAL   DEV_TYP_STRNG,R8        ; Get address of device type strings.
67   01     D1  CMPL    #DT$_CI780,(R7)             ; Is the port a CI780?  ********
     0C     13  BEQL    10$                         ; Branch if so.
58   88     80  ADDB    (R8)+,R8                    ; Otherwise point to the next string.
67   02     D1  CMPL    #DT$_CI750,(R7)             ; Is the port a CI750?  ********
     04     13  BEQL    10$                         ; Branch if so.
     50     D4  CLRL    R0                          ; Otherwise indicate failure.
     06     11  BRB     99$                         ; Any return imediately.
57   58     D0  MOVL    R8,R7                       ; Get address of the string in R7.
   F89B     30  BSBW    FORMAT_DATA                 ; Format the string.
0180 8F     BA  POPR    #^M<R7,R8>
             05  RSB
```

K 1

2CE9 8839

```
                                                     L 1
2CE9  8841 ;++
2CE9  8842 ;
2CE9  8843 ; FUNCTIONAL DESCRIPTION:
2CE9  8844 ;
2CE9  8845 ;       This routine is used to display the loopback status of the port.
2CE9  8846 ;       The value at offsets PDT$B_P0_LBSTS and PDT$B_P1_LBSTS are checked for
2CE9  8847 ;       loopback or normal status and converted to an ascii string as follows:
2CE9  8848 ;
2CE9  8849 ;       "A - B"         -       Loopback tests passed on paths A and B.
2CE9  8850 ;       "A - "          -       Loopback tests passed on path A.
2CE9  8851 ;       "  - B"         -       Loopback tests passed on path B.
2CE9  8852 ;       "  - "          -       Loopback tests failed on paths A and B.
2CE9  8853 ;       " N/A "         -       Loopback testing is not being done.
2CE9  8854 ;
2CE9  8855 ;       The ascii string is stored in the format buffer.
2CE9  8856 ;
2CE9  8857 ; CALLING SEQUENCE:
2CE9  8858 ;
2CE9  8859 ;       JSB     DISPLAY_LOOPBACK_STATUS
2CE9  8860 ;
2CE9  8861 ; CALLED BY:
2CE9  8862 ;
2CE9  8863 ;       GET_DATA
2CE9  8864 ;
2CE9  8865 ; INPUTS:
2CE9  8866 ;
2CE9  8867 ;       R7      = Offset PDT$B_P0_LBSTS from DEFINE_FIELD macro.
2CE9  8868 ;       (AP)[PDT_IN]    = Base address of the data for selected port in PDT_BUF.
2CE9  8869 ;
2CE9  8870 ; IMPLICIT INPUTS:
2CE9  8871 ;
2CE9  8872 ;       AP      = Address of call frame containing the addresses of the data
2CE9  8873 ;                 structures to be used when data is extracted.
2CE9  8874 ;
2CE9  8875 ; OUTPUTS:
2CE9  8876 ;
2CE9  8877 ;       Formated ascii string returned in FORMAT_BUF.
2CE9  8878 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
2CE9  8879 ;
2CE9  8880 ; RETURN STATUS:
2CE9  8881 ;
2CE9  8882 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
2CE9  8883 ;
2CE9  8884 ;--
2CE9  8885
2CE9  8886 ASSUME  PDT$B_P1_LBSTS  EQ PDT$B_P0_LBSTS+1
00000002  2CE9  8887 PDT$V_LBDG      =       2
2CE9  8888
2CE9  8889 DISPLAY_LOOPBACK_STATUS:
0380 8F   BB  2CE9  8890         PUSHR   #^M<R7,R8,R9>
58   06   D0  2CED  8891         MOVL    #PDT_IN,R8                      ; Set index into call frame to base
2CF0  8892                                                               ;  address of PDT for this port.
57  00000110 8F   6C48   C1  2CF0  8893         ADDL3   (R12)[R8],-             ; Get address of local port status.
2CF9  8894                 #PDT$W_LPORT_STS,R7             ;
57  FFFFFF4C 8F   C0  2CF9  8895         ADDL    #SHC$L_PDT_DIF,R7       ; Adjust PDT offset to compensate for
2D00  8896                                                               ;  portion of PDT not copied.
003A   30  2D00  8897         BSBW    CHECK_DIF                       ; Check for change in data since last
```

```
                                2D03  8898
                                      ;  update.
            05 57    02     E0  2D03  8899          BBS     #PDT$V_LBDG,R7,5$       ; Branch if loopback test enabled.
               59    04     D0  2D07  8900          MOVL    #4,R9                   ; Set index to point to "N/A" string.
                     1F     11  2D0A  8901          BRB     20$                     ; Display the string.
                     59     D4  2D0C  8902 5$:      CLRL    R9                      ; Assume loopback tests succeeded on
                                2D0E  8903                                          ;  both ports.
      57    00000070 8F     C0  2D0E  8904          ADDL    #<PDT$B_PO_LBSTS--      ; Get address of path A loopback
                                2D15  8905                  PDT$W_LPORT_STS>,R7     ;  status.
                  0025     30  2D15  8906          BSBW    CHECK_DIF               ; Check for change in path A loopback
                                2D18  8907                                          ;  status.
                  87    02     93  2D18  8908          BITB    #PDT$M_PRV_LBS,(R7)+    ; Path A loopback test succeeded?
                        03     12  2D1B  8909          BNEQ    10$                     ; Branch if so.
                  59    02     C8  2D1D  8910          BISL    #2,R9                   ; Index = index + 2.
                     001A     30  2D20  8911 10$:     BSBW    CHECK_DIF               ; Check for change in path B loopback
                                2D23  8912                                          ;  status.
                  67    02     93  2D23  8913          BITB    #PDT$M_PRV_LBS,(R7)     ; Path B loopback test suceeded?
                        03     12  2D26  8914          BNEQ    20$                     ; Branch if so.
                  59    01     C8  2D28  8915          BISL    #1,R9                   ; Index = index + 1.
                  57    59     D0  2D2B  8916 20$:     MOVL    R9,R7                   ; Get index in R7 for DISPLAY_STRING.
      58    000002AC'EF     DE  2D2E  8917          MOVAL   LB_STAT_STRG,R8         ; Point to loopback status strings.
                  FCAE     30  2D35  8918          BSBW    DISPLAY_STRING          ; Convert index to string and format.
               0380 8F     BA  2D38  8919          POPR    #^M<R7,R8,R9>
                           05  2D3C  8920          RSB
                               2D3D  8921
                               2D3D  8922
```

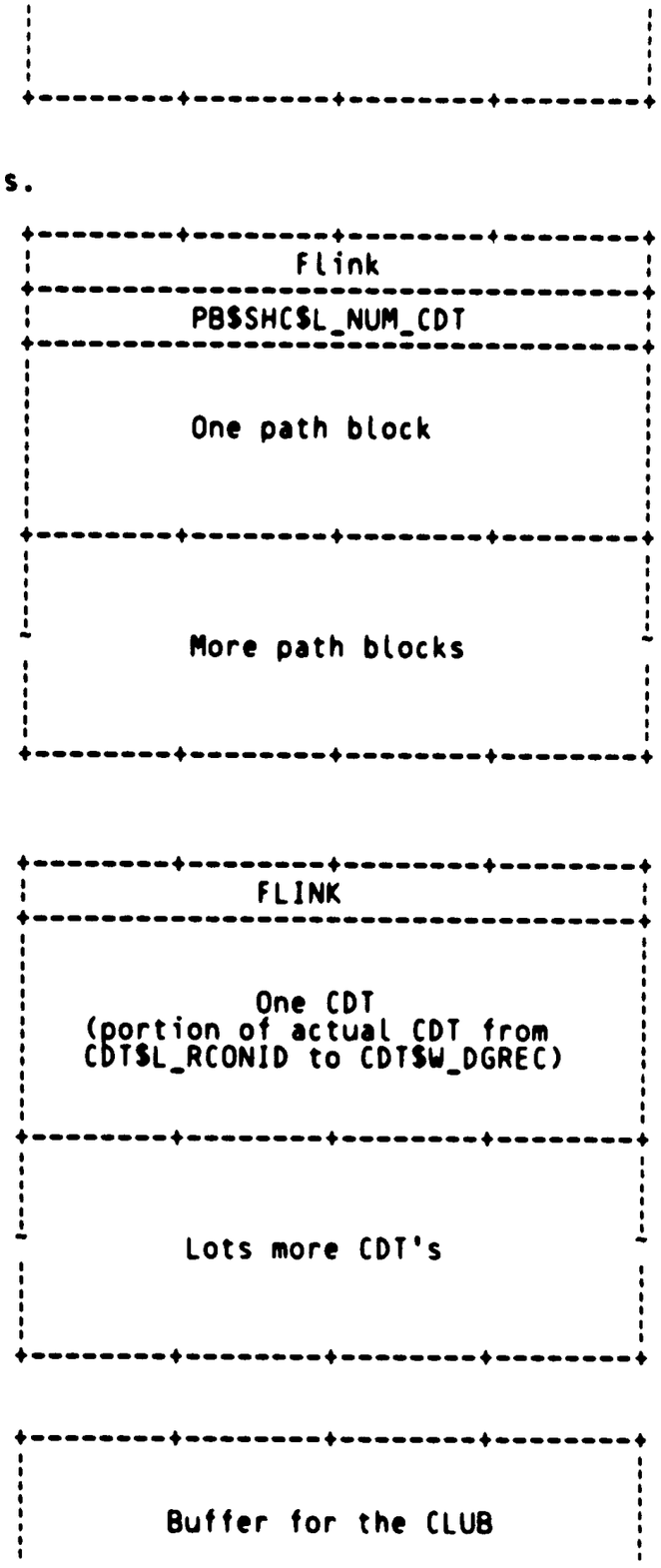
```
                                N 1
2D3D  8924          .SBTTL  -       CHECK_DIF routine.
2D3D  8925 ;++
2D3D  8926 ;
2D3D  8927 ; FUNCTIONAL DESCRIPTION:
2D3D  8928 ;
2D3D  8929 ;       This routine uses the address of the data in R7 to determine which
2D3D  8930 ;       buffer the data is to be extracted from.  It the uses that address
2D3D  8931 ;       to determine where the matching data in the corresponding buffer from
2D3D  8932 ;       the previous snapshot can be found.  The data is then compared and
2D3D  8933 ;       if found to be is different then a flag is set.  The flag is used by
2D3D  8934 ;       the GET_DATA routine to determine when to turn on inverse video.
2D3D  8935 ;
2D3D  8936 ; CALLING SEQUENCE:
2D3D  8937 ;
2D3D  8938 ;       JSB     CHECK_DIF
2D3D  8939 ;
2D3D  8940 ; CALLED BY:
2D3D  8941 ;
2D3D  8942 ;       DISPLAY_PORT_TYPE
2D3D  8943 ;       DISPLAY_PORT_STATUS
2D3D  8944 ;       DISPLAY_SCS_STATE
2D3D  8945 ;       DISPLAY_CON_STATE
2D3D  8946 ;       DISPLAY_PROC_NAME
2D3D  8947 ;       DISPLAY_CABLE_STATUS
2D3D  8948 ;       DISPLAY_RSTATE
2D3D  8949 ;       DISPLAY_CIR_STATE
2D3D  8950 ;       DISPLAY_INCN_TIME
2D3D  8951 ;       DISPLAY_FIELD
2D3D  8952 ;
2D3D  8953 ; INPUTS:
2D3D  8954 ;
2D3D  8955 ;       Address of the data from the most recent snapshot in R7.
2D3D  8956 ;
2D3D  8957 ; IMPLICIT INPUTS:
2D3D  8958 ;
2D3D  8959 ;       Data from the previous snapshot stored in OLD_VAR_BUF and OLD_FIX_BUF.
2D3D  8960 ;
2D3D  8961 ; OUTPUTS:
2D3D  8962 ;
2D3D  8963 ;       Flag set in MISC_FLAGS if new data is different from the old data.
2D3D  8964 ;
2D3D  8965 ; RETURN STATUS:
2D3D  8966 ;
2D3D  8967 ;       None.
2D3D  8968 ;
2D3D  8969 ; REGISTER USAGE:
2D3D  8970 ;
2D3D  8971 ;       R2,R3 - Used to calculate the address of the data in the old buffer
2D3D  8972 ;       R6 -    Receives the number of bytes of data in the field.  Used to
2D3D  8973 ;               to compare the new data with the old.
2D3D  8974 ;       R7 -    Contains the address of the data.  Used to calculate the
2D3D  8975 ;               address of the data in the old snapshot buffer.
2D3D  8976 ;--
2D3D  8977
2D3D  8978 CHECK_DIF:
004C 8F   BB  2D3D  8979          PUSHR   #^M<R2,R3,R6>
000004A3'8F   57  D1  2D41  8980          CMPL    R7,#SNAP_BUF_END        ; Was the data extracted from the fixed
```

```
                                2D48  8981
                                      ;  length buffers.
                     71   1F    2D48  8982          BLSSU   40$                     ; Branch if so.
00000033'EF          57   D1    2D4A  8983          CMPL    R7,CSB_DSCR+4           ; Was the data extracted from the
                                2D51  8984                                          ;  system block buffer?
                     10   1E    2D51  8985          BGEQU   10$                     ; Branch if not.
52   0000002B'EF     D0   2D53  8986          MOVL    SB_DSCR+4,R2            ; Get the address of the system block
                                2D5A  8987                                          ;  buffer.
53   000003FD'EF     D0   2D5A  8988          MOVL    OLD_SB_DSCR+4,R3        ; Get address of system block buffer
                                2D61  8989                                          ;  from previous snapshot.
                     66   11    2D61  8990          BRB     50$                     ; Jump to common code.
0000003B'EF          57   D1    2D63  8991 10$:     CMPL    R7,PB_DSCR+4            ; Was the data extracted from the
                                2D6A  8992                                          ;  cluster system block buffer?
                     10   1E    2D6A  8993          BGEQU   20$                     ; Branch if not.
52   00000033'EF     D0   2D6C  8994          MOVL    CSB_DSCR+4,R2           ; Get the address of the cluster system
                                2D73  8995                                          ;  block buffer.
53   00000405'EF     D0   2D73  8996          MOVL    OLD_CSB_DSCR+4,R3       ; Get address of cluster system block
                                2D7A  8997                                          ;  buffer from previous snapshot.
                     4D   11    2D7A  8998          BRB     50$                     ; Jump to common code.
00000043'EF          57   D1    2D7C  8999 20$:     CMPL    R7,CDT_DSCR+4           ; Was the data extracted from the
                                2D83  9000                                          ;  path block buffer?
                     10   1E    2D83  9001          BGEQU   30$                     ; Branch if not.
52   0000003B'EF     D0   2D85  9002          MOVL    PB_DSCR+4,R2            ; Get the address of the path block
                                2D8C  9003                                          ;  buffer.
53   0000040D'EF     D0   2D8C  9004          MOVL    OLD_PB_DSCR+4,R3        ; Get address of path block buffer
                                2D93  9005                                          ;  from previous snapshot.
                     34   11    2D93  9006          BRB     50$                     ; Jump to common code.
56   00000043'EF     D0   2D95  9007 30$:     MOVL    CDT_DSCR+4,R6           ; Get the starting address of the
                                2D9C  9008                                          ;  CDT buffer.
53   0000003F'EF     3C   2D9C  9009          MOVZWL  CDT_DSCR,R3             ; Get the length of the CDT buffer.
          56   53     C0   2DA3  9010          ADDL    R3,R6                   ; Calculate ending address of CDT
                                2DA6  9011                                          ;  buffer.
          56   57     D1   2DA6  9012          CMPL    R7,R6                   ; Was the data extracted from the
                                2DA9  9013                                          ;  CDT buffer?
                     36   1E    2DA9  9014          BGEQU   99$                     ; Branch if not.
52   00000043'EF     D0   2DAB  9015          MOVL    CDT_DSCR+4,R2           ; Get the address of the CDT buffer.
53   00000415'EF     D0   2DB2  9016          MOVL    OLD_CDT_DSCR+4,R3       ; Get address of CDT buffer from
                                2DB9  9017                                          ;  previous snapshot.
                     0E   11    2DB9  9018          BRB     50$                     ; Jump to common code.
52   0000004B'EF     DE   2DBB  9019 40$:     MOVAL   FIX_BUF,R2              ; Get the starting address of the
                                2DC2  9020                                          ;  fixed length buffers.
                                2DC2  9021
53   000004A3'EF     DE   2DC2  9022          MOVAL   OLD_FIX_BUF,R3          ; Get the starting address of the fixed
                                2DC9  9023                                          ;  length buffers from the previous
                                2DC9  9024                                          ;  snapshot.
     52   57   52     C3   2DC9  9025 50$:     SUBL3   R2,R7,R2                ; Get the offset into the buffer where
                                2DCD  9026                                          ;  the data is located.
          53   52     C0   2DCD  9027          ADDL    R2,R3                   ; Get the corresponding address of the
                                2DD0  9028                                          ;  data from the previous snapshot.
       56   04 AA     9B   2DD0  9029          MOVZBW  FD$B_LENGTH(R10),R6     ; Get the number of bytes of raw data.
     63   67   56     29   2DD4  9030          CMPC    R6,(R7),(R3)            ; Compare the old data with the new.
                     07   13    2DD8  9031          BEQL    99$                     ; Branch if they're the same.
00000476'EF          04   C8    2DDA  9032          BISL    #MF$M_INVERSE,MISC_FLAGS; Set the inverse video flag.
          004C 8F    BA   2DE1  9033 99$:     POPR    #^M<R2,R3,R6>
                     05   2DE5  9034          RSB
                                2DE6  9035
                                2DE6  9036
```

```
2DE6  9038          .SBTTL  Description of snapshot buffers.
2DE6  9039 ;
2DE6  9040 ;SNAP_BUF:       Start of buffers to be locked.
2DE6  9041 ;
2DE6  9042 ;                            +---------+
2DE6  9043 ;CURRENT_REPORT:             |         |   Bit 0 set indicates cluster report.
2DE6  9044 ;                            +---------+
2DE6  9045 ;
2DE6  9046 ;                            +--------+--------+--------+--------+
2DE6  9047 ;CLASS_SELECT:               | USER selected classes for display |
2DE6  9048 ;                            +--------+--------+--------+--------+
2DE6  9049 ;
2DE6  9050 ;                            +--------+--------+--------+--------+--------+--------+
2DE6  9051 ;LOCSYSID:                   |           ID number of the local system.          |
2DE6  9052 ;                            +--------+--------+--------+--------+--------+--------+
2DE6  9053 ;
2DE6  9054 ;                            +--------+--------+--------+--------+
2DE6  9055 ;NUM_BUF_DSCR:               |  # of buffer descriptors in use   |
2DE6  9056 ;                            +--------+--------+--------+--------+
2DE6  9057 ;
2DE6  9058 ;                            +--------+--------+--------+--------+
2DE6  9059 ;NUM_FREE_BUF_DSCR:          | # of buffer descriptors available |
2DE6  9060 ;                            +--------+--------+--------+--------+
2DE6  9061 ;
2DE6  9062 ;                            +--------+--------+--------+--------+
2DE6  9063 ;LOC_SB_ADRS:                |   Address of local system block   |
2DE6  9064 ;                            +--------+--------+--------+--------+
2DE6  9065 ;
2DE6  9066 ;                            +--------+--------+--------+--------+
2DE6  9067 ;SB_BUF_PNTR:                |       Pointer into SB buffer      |
2DE6  9068 ;                            +--------+--------+--------+--------+
2DE6  9069 ;                            +--------+--------+--------+--------+
2DE6  9070 ;CSB_BUF_PNTR:               |       Pointer into CSB buffer     |
2DE6  9071 ;                            +--------+--------+--------+--------+
2DE6  9072 ;                            +--------+--------+--------+--------+
2DE6  9073 ;PB_BUF_PNTR:                |       Pointer into PB buffer      |
2DE6  9074 ;                            +--------+--------+--------+--------+
2DE6  9075 ;                            +--------+--------+--------+--------+
2DE6  9076 ;CDT_BUF_PNTR:               |       Pointer into CDT buffer     |
2DE6  9077 ;                            +--------+--------+--------+--------+
2DE6  9078 ;
2DE6  9079 ;                            +--------+--------+--------+--------+
2DE6  9080 ; VAR_BUF_ADRS:              |Address of variable length buffers |
2DE6  9081 ;                            +--------+--------+--------+--------+
2DE6  9082 ;
2DE6  9083 ; Descriptors for the variable length snapshot buffers.
2DE6  9084 ;
2DE6  9085 ;                            +--------+--------+--------+--------+
2DE6  9086 ;SB_DSCR:                    |#entrys in buffer|Length of buffer |
2DE6  9087 ;                            +--------+--------+--------+--------+
2DE6  9088 ;                            |         Address of buffer         |
2DE6  9089 ;                            +--------+--------+--------+--------+
2DE6  9090 ;
2DE6  9091 ;                            +--------+--------+--------+--------+
2DE6  9092 ;CSB_DSCR:                   |#entrys in buffer|Length of buffer |
2DE6  9093 ;                            +--------+--------+--------+--------+
2DE6  9094 ;                            |         Address of buffer         |
```

```
2DE6  9095 ;                          +--------+--------+--------+--------+
2DE6  9096 ;
2DE6  9097 ;                          +--------+--------+--------+--------+
2DE6  9098 ;PB_DSCR:                  |#entrys in buffer|Length of buffer |
2DE6  9099 ;                          +--------+--------+--------+--------+
2DE6  9100 ;                          |         Address of buffer         |
2DE6  9101 ;                          +--------+--------+--------+--------+
2DE6  9102 ;
2DE6  9103 ;                          +--------+--------+--------+--------+
2DE6  9104 ;CDT_DSCR:                 |#entrys in buffer|Length of buffer |
2DE6  9105 ;                          +--------+--------+--------+--------+
2DE6  9106 ;                          |         Address of buffer         |
2DE6  9107 ;                          +--------+--------+--------+--------+
2DE6  9108 ;
2DE6  9109 ; Buffer for system blocks.
2DE6  9110 ;
2DE6  9111 ;                          +--------+--------+--------+--------+
2DE6  9112 ;          ^ SB$L_FLINK    |   Pointer to next system block    |
2DE6  9113 ;          |               +-----------------------------------+
2DE6  9114 ;          | SB$L_BLINK:   |          SB$SHC$L_NUM_PB          |
2DE6  9115 ;          |               +-----------------+--------+--------+
2DE6  9116 ;          |               |                 |SHC$SB$B|        |
2DE6  9117 ;          |               |                 |_DIS_POS|        |
2DE6  9118 ;          |               +-----------------+--------+--------+
2DE6  9119 ;    SB$C_LENGTH           |                                   |
2DE6  9120 ;          |               |                                   |
2DE6  9121 ;          |               |                                   |
2DE6  9122 ;          |               |                                   |
2DE6  9123 ;          |               |                                   |
2DE6  9124 ;          |               +--------+--------+--------+--------+
2DE6  9125 ;          V    SB$L_CSB:  |  Pointer to CSB for this system   |
2DE6  9126 ;               +--------+--------+--------+--------+
2DE6  9127 ;               |                                   |
2DE6  9128 ;               |                                   |
2DE6  9129 ;               |                                   |
2DE6  9130 ;               r~       More system blocks         |
2DE6  9131 ;               |                                   |
2DE6  9132 ;               |                                   |
2DE6  9133 ;               |                                   |
2DE6  9134 ;               +--------+--------+--------+--------+
2DE6  9135 ;
2DE6  9136 ;
2DE6  9137 ; Buffer for cluster system blocks.
2DE6  9138 ;
2DE6  9139 ;                          +--------+--------+--------+--------+
2DE6  9140 ;                          ^                                   |
2DE6  9141 ;                          |                                   |
2DE6  9142 ;                          |                                   |
2DE6  9143 ;          CSB$C_LENGTH    |    One cluster system block       |
2DE6  9144 ;                          |                                   |
2DE6  9145 ;                          |                                   |
2DE6  9146 ;                          V                                   |
2DE6  9147 ;                          +--------+--------+--------+--------+
2DE6  9148 ;                          |                                   |
2DE6  9149 ;                          |                                   |
2DE6  9150 ;                          |                                   |
2DE6  9151 ;                          ~    More cluster system blocks     |
```

```
2DE6  9152 ;
2DE6  9153 ;
2DE6  9154 ;
2DE6  9155 ;
2DE6  9156 ;
2DE6  9157 ;
2DE6  9158 ; Buffer for path blocks.
2DE6  9159 ;
2DE6  9160 ;
2DE6  9161 ;                    ^
2DE6  9162 ;                    |
2DE6  9163 ;                    |
2DE6  9164 ;                    |
2DE6  9165 ;            PB$C_LENGTH
2DE6  9166 ;                    |
2DE6  9167 ;                    |
2DE6  9168 ;                    |
2DE6  9169 ;                    |
2DE6  9170 ;                    V
2DE6  9171 ;
2DE6  9172 ;
2DE6  9173 ;
2DE6  9174 ;
2DE6  9175 ;
2DE6  9176 ;
2DE6  9177 ;
2DE6  9178 ;
2DE6  9179 ;
2DE6  9180 ;
2DE6  9181 ; Buffer for CDT's.
2DE6  9182 ;
2DE6  9183 ;
2DE6  9184 ;                    ^
2DE6  9185 ;                    |
2DE6  9186 ;                    |
2DE6  9187 ;
2DE6  9188 ;      SHC$C_CDT_LENGTH+32
2DE6  9189 ;
2DE6  9190 ;                    |
2DE6  9191 ;                    |
2DE6  9192 ;                    V
2DE6  9193 ;
2DE6  9194 ;
2DE6  9195 ;
2DE6  9196 ;
2DE6  9197 ;
2DE6  9198 ;
2DE6  9199 ;
2DE6  9200 ;
2DE6  9201 ;
2DE6  9202 ;
2DE6  9203 ; Fixed length buffers
2DE6  9204 ;
2DE6  9205 ;CLUB_BUF:          ^
2DE6  9206 ;                    |
2DE6  9207 ;            CLUB$C_LENGTH
2DE6  9208 ;                    |
```



```
2DE6  9209 ;                    |                 |
2DE6  9210 ;                    v                 +-------+-------+-------+-------+
2DE6  9211 ;                                      +-------+-------+-------+-------+
2DE6  9212 ;DUMMY_SB:           ^                 |                               |
2DE6  9213 ;                    |                 |                               |
2DE6  9214 ;                    |                 |                               |
2DE6  9215 ;                                      |                               |
2DE6  9216 ;                SB$C_LENGTH           |       One fake system block   |
2DE6  9217 ;                                      |                               |
2DE6  9218 ;                    |                 |                               |
2DE6  9219 ;                    |                 |                               |
2DE6  9220 ;                    V                 |                               |
2DE6  9221 ;                                      +-------+-------+-------+-------+
2DE6  9222 ;DUMMY_PB:           ^                 |                               |
2DE6  9223 ;                    |                 |                               |
2DE6  9224 ;                    |                 |                               |
2DE6  9225 ;                                      |                               |
2DE6  9226 ;                PB$C_LENGTH           |       One fake path block     |
2DE6  9227 ;                                      |                               |
2DE6  9228 ;                    |                 |                               |
2DE6  9229 ;                    |                 |                               |
2DE6  9230 ;                    V                 |                               |
2DE6  9231 ;                                      +-------+-------+-------+-------+
2DE6  9232 ;
2DE6  9233 ; Buffer for PDT's.
2DE6  9234 ;
2DE6  9235 ;                                      +-------+-------+-------+-------+
2DE6  9236 ;PDT_BUF: ^          ^                |                               |
2DE6  9237 ;         |          |                |                               |
2DE6  9238 ;         |          |                |                               |
2DE6  9239 ;                    |                |          First PDT            |
2DE6  9240 ; SHC$L_PDT_LENGTH   |                |  (portion of actual PDT from  |
2DE6  9241 ;                    |                | PDT$L_MSGHDRSZ to PDT$L_LBDG) |
2DE6  9242 ;                    |                |                               |
2DE6  9243 ;         |          |                |                               |
2DE6  9244 ;         V          |                |                               |
2DE6  9245 ;                                      +-------+-------+-------+-------+
2DE6  9246 ;                    |                |                               |
2DE6  9247 ;                    |                |                               |
2DE6  9248 ;SHC$L_PDT_LENGTH*MAXPDT              |                               |
2DE6  9249 ;                    |                |                               |
2DE6  9250 ;                    |                |          Second PDT           |
2DE6  9251 ;                    |                |                               |
2DE6  9252 ;                    |                |                               |
2DE6  9253 ;                    |                |                               |
2DE6  9254 ;                    V                |                               |
2DE6  9255 ;                                      +-------+-------+-------+-------+
2DE6  9256 ;
2DE6  9257 ;
2DE6  9258 ; Buffer for miscellaneous port data.
2DE6  9259 ;
2DE6  9260 ;
2DE6  9261 ;                                      +-------+-------+-------+-------+
2DE6  9262 ;MISC_BUF:                             |          MB$L_PORT_NAME       |
2DE6  9263 ;                                      +-------+-------+-------+-------+
2DE6  9264 ;                                      |          MB$L_PORT_NUM        |
2DE6  9265 ;                                      +-------+-------+-------+-------+
```

```
2DE6  9266 ;                          +-------+-------+-------+-------+
2DE6  9267 ;                          |           MB$L_DEVTYPE        |
2DE6  9268 ;                          +-------+-------+-------+-------+
2DE6  9269 ;                          |           MB$L_STS            |
2DE6  9270 ;                          +-------+-------+-------+-------+
2DE6  9271 ;                          | MB$W_ERRCNT   | ERTMAX | ERTCNT |
2DE6  9272 ;                          +-------+-------+-------+-------+
2DE6  9273 ;                          |           MB$L_FORM_PB        |
2DE6  9274 ;                          +-------+-------+-------+-------+
2DE6  9275 ;                          |           MB$L_PL_WAITERS     |
2DE6  9276 ;                          +-------+-------+-------+-------+
2DE6  9277 ;                          | MB$W_MSG_FREE | MB$W_DG_FREE  |
2DE6  9278 ;                          +-------+-------+-------+-------+
2DE6  9279 ;                          | MB$W_RSPS_QUED | MB$W_CMDS_QUED |
2DE6  9280 ;                          +-------+-------+-------+-------+
2DE6  9281 ;MISC_BUF+MISC_BUF_LEN:    |                               |
2DE6  9282 ;                          |                               |
2DE6  9283 ;                          |                               |
2DE6  9284 ;                          |                               |
2DE6  9285 ;                          |                               |
2DE6  9286 ;                          |     MISC_BUF for second port  |
2DE6  9287 ;                          |                               |
2DE6  9288 ;                          |                               |
2DE6  9289 ;                          |                               |
2DE6  9290 ;                          |                               |
2DE6  9291 ;                          |                               |
2DE6  9292 ;                          |                               |
2DE6  9293 ;                          +-------+-------+-------+-------+
2DE6  9294 ;
2DE6  9295 ;
2DE6  9296 ;SNAP_BUF_END:             End of buffers to be locked.
2DE6  9297 ;
2DE6  9298 ;
2DE6  9299
2DE6  9300
```

```
                     2DE6  9302          .SBTTL  Snapshot buffers.
                 00000000  9303          .PSECT  SNAP_DATA,RD,WRT,NOEXE
                     0000  9304
                     0000  9305  SNAP_BUF:                          ; Start of buffers to be locked.
                     0000  9306
                     0000  9307  CURRENT_REPORT:                    ; Indicates which report the user has
                       00  0000  9308          .BYTE   0            ;  selected for display.
                     0001  9309
                     0001  9310  CLASS_SELECT:                     ; Indicates which classes the user has
                 00000000  0001  9311          .LONG   0            ;  selected for display.
                     0005  9312
                     0005  9313  LOCSYSID:                         ; ID number of the local system.
                 0000000B  0005  9314          .BLKB   6
                     000B  9315
                     000B  9316  NUM_BUF_DSCR:                     ; Number of buffer descriptors in use.
                 00000000  000B  9317          .LONG   0
                     000F  9318
                     000F  9319  NUM_FREE_BUF_DSCR:                ; Number of buffer descriptors
                 00000000  000F  9320          .LONG   0            ;  available.
                     0013  9321
                     0013  9322  LOC_SB_ADRS:                      ; Address of system block for local
                 00000000  0013  9323          .LONG   0            ;  system.
                     0017  9324
                     0017  9325  SB_BUF_PNTR:                      ; Pointer into system block buffer.
                 00000000  0017  9326          .LONG   0            ;
                     001B  9327  CSB_BUF_PNTR:                     ; Pointer into cluster system block
                 00000000  001B  9328          .LONG   0            ;  buffer.
                     001F  9329  PB_BUF_PNTR:                      ; Pointer in path block buffer.
                 00000000  001F  9330          .LONG   0            ;
                     0023  9331  CDT_BUF_PNTR:                     ; Pointer into CDT buffer.
                 00000000  0023  9332          .LONG   0            ;
                     0027  9333
 00000000 00000000   0027  9334  SB_DSCR:.QUAD   0                 ; System block buffer.
                     002F  9335
                     002F  9336  CSB_DSCR:                         ; Cluster system block buffer.
 00000000 00000000   002F  9337          .QUAD   0                ;
                     0037  9338
 00000000 00000000   0037  9339  PB_DSCR:.QUAD   0                 ; Path block buffer.
                     003F  9340
                     003F  9341  CDT_DSCR:                         ; Connection descriptor table buffer.
 00000000 00000000   003F  9342          .QUAD   0                ;
                     0047  9343
                     0047  9344  VAR_BUF_ADRS:                     ; Copy of the variable buffer address
                 00000000  0047  9345          .LONG   0            ;  returned from GET_VM.
                     004B  9346
                     004B  9347  FIX_BUF:                          ; Start of fixed length buffers.
                     004B  9348
                     004B  9349  CLUB_BUF:                         ; Buffer for the club.
                 000001F3  004B  9350          .BLKB   CLUB$C_LENGTH ;
                     01F3  9351
                     01F3  9352  DUMMY_SB:                         ; Fake system block for formative path
                 00000253  01F3  9353          .BLKB   SB$C_LENGTH  ;  blocks without an associated
                     0253  9354                                     ;  formative system block.
                     0253  9355
                     0253  9356  DUMMY_PB:                         ; Fake path block for connections in
                 000002A7  0253  9357          .BLKB   PB$C_LENGTH  ;  the listen state.
                     02A7  9358
```

```
                    02A7  9359 PDT_BUF:
00000447            02A7  9360          .BLKB   SHC$L_PDT_LENGTH*MAXPDT ; Buffer for port information.
                    0447  9361
                    0447  9362 MISC_BUF:                                ; Buffer for miscellaneous information.
000004A3            0447  9363          .BLKB   MISC_BUF_LEN*2
                    04A3  9364
                    04A3  9365 SNAP_BUF_END:                           ; End of buffers to be locked.
                    04A3  9366
00000458            04A3  9367 FIX_BUF_LEN      = .-FIX_BUF
000004A3            04A3  9368 SNAP_BUF_LEN     = .-SNAP_BUF
                    04A3  9369
                    04A3  9370
```

```
                04A3  9372          .SBTTL  Snapshot routines.
          0000 2DE6  9373          .PSECT  MAIN,RD,EXE,NOWRT
               2DE6  9374 ;++
               2DE6  9375 ;
               2DE6  9376 ; FUNCTIONAL DESCRIPTION:
               2DE6  9377 ;
               2DE6  9378 ;       This routine is used to extract the data from the SCS database and the
               2DE6  9379 ;       CI port database.  This routine is called whenever the interval timer
               2DE6  9380 ;       expires or upon completion of parsing and executing a command.
               2DE6  9381 ;
               2DE6  9382 ;
               2DE6  9383 ; CALLING SEQUENCE:
               2DE6  9384 ;
               2DE6  9385 ;       $CMKRNL_S -
               2DE6  9386 ;               ROUTINE=SNAPSHOT
               2DE6  9387 ;
               2DE6  9388 ; CALLED BY:
               2DE6  9389 ;
               2DE6  9390 ;       DO_SNAPSHOT
               2DE6  9391 ;
               2DE6  9392 ;
               2DE6  9393 ; IMPLICIT IMPUTS:
               2DE6  9394 ;
               2DE6  9395 ;       Cluster and port data stored in SCS database and CI port database.
               2DE6  9396 ;
               2DE6  9397 ;
               2DE6  9398 ; IMPLICIT OUTPUTS:
               2DE6  9399 ;
               2DE6  9400 ;       Extracted data stored in snapshot buffers.  For details see previous
               2DE6  9401 ;       section.
               2DE6  9402 ;
               2DE6  9403 ;
               2DE6  9404 ; RETURN STATUS:
               2DE6  9405 ;
               2DE6  9406 ;       SS$_NORMAL or SS$_NOSUCHNODE returned in R0.
               2DE6  9407 ;
               2DE6  9408 ;
               2DE6  9409 ; SIDE EFFECTS:
               2DE6  9410 ;
               2DE6  9411 ;       It should be noted that this routine runs at IPL$_SCS and calls a
               2DE6  9412 ;       routine CHECK_QUE that runs at IPL 31.  Also both routines and all
               2DE6  9413 ;       the buffers used by the routines are locked in memory during execution
               2DE6  9414 ;       of this routine.
               2DE6  9415 ;
               2DE6  9416 ;
               2DE6  9417 ; REGISTER USAGE:
               2DE6  9418 ;
               2DE6  9419 ;       Listed here is a general register usage.  Detailed usage is given in
               2DE6  9420 ;       each subsection.
               2DE6  9421 ;
               2DE6  9422 ;       R3  -   Used as destination pointer when storing structures in buffers.
               2DE6  9423 ;       R4  -   Used as a temporary register and by MOVC's.
               2DE6  9424 ;       R5  -   Used as a temporary register and by MOVC's.
               2DE6  9425 ;       R6  -   Used as a temporary register.
               2DE6  9426 ;       R7  -   Used to store the address of the first link in a doubly linked
               2DE6  9427 ;               list.
               2DE6  9428 ;       R8  -   Used as a temporary register and pointer to the base address to
```

```
                                2DE6  9429 ;               store data in MISC_BUF.
                                2DE6  9430 ;       R9  -   Used as a temporary register and to save a copy of R3 before
                                2DE6  9431 ;               R3 is updated to point to next slot in buffer.
                                2DE6  9432 ;       R10 -   Used to store the address of the next link in a doubly linked
                                2DE6  9433 ;               list.
                                2DE6  9434 ;           -   Used as source pointer when storing structures in buffers.
                                2DE6  9435 ;--
                                2DE6  9436
                          07F8  2DE6  9437 .ENTRY  SNAPSHOT,^M<R3,R4,R5,R6,R7,R8,R9,R10>
                            00  2DE8  9438          BBC     #CLS$V_CLUB,-           ; Branch if display of the club is
                    16 00000001'EF  2DEA  9439                  CLASS_SELECT,5$         ;  disabled.
                5A  00000000'GF  DO  2DF0  9440          MOVL    G^CLU$GL_CLUB,R10       ; Get address of the club.
                            0D  13  2DF7  9441          BEQL    5$                      ; Branch if there isn't a club.
                53  0000004B'EF  DE  2DF9  9442          MOVAL   CLUB_BUF,R3             ; Get address to store the club.
                    6A  01A8 8F  28  2E00  9443          MOVC3   #CLUB$C_LENGTH,(R10),-  ; Store club in club buffer.
                            63        2E05  9444                  (R3)                    ;
                50  0000028C 8F  DO  2E06  9445 5$:      MOVL    #SS$_NOSUCHNODE,R0      ; Assume that there is no cluster.
                    00000000'GF  D1  2E0D  9446          CMPL    G^SCS$GQ_CONFIG,-       ; Any remote SB's known?
                    00000000'8F      2E13  9447                  #SCS$GQ_CONFIG          ;
                                     2E18  9448          JEQL    SNAPSHOT_ERR            ; If not, exit with error status.
                                     2E1D  9449 10$:     DSBINT  #IPL$_SCS               ; Lock the SCS database.
                            00  E0  2E23  9450          BBS     #RPT$V_CLSTR,-          ; Branch to cluster snapshot routine
                    00000000'EF      2E25  9451                  CURRENT_REPORT,-        ;  if we're displaying the cluster
                            03       2E2A  9452                  CLUSTER_SNAP            ;  report.
                          02B1  31  2E2B  9453          BRW     LOCAL_SNAP              ; Otherwise branch to the local
                                     2E2E  9454                                          ;  snapshot routine.
                                     2E2E  9455
                                     2E2E  9456 ;
                                     2E2E  9457 ; Store the information necessary to display the cluster report.  This includes
                                     2E2E  9458 ; all system blocks, path blocks, and connection descriptors.  Formative path
                                     2E2E  9459 ; blocks and connections descriptors for connections in the listen state will
                                     2E2E  9460 ; also be stored if the user has elected to see them.
                                     2E2E  9461 ;
                                     2E2E  9462
                                     2E2E  9463 CLUSTER_SNAP:
                    00000013'EF  D4  2E2E  9464          CLRL    LOC_SB_ADRS             ; Clear address of local system block.
                    000001FF'EF  D4  2E34  9465          CLRL    DUMMY_SB+SB$L_PBFL      ; Assume no formative path blocks with
                                     2E3A  9466                                          ;  formative system blocks.
                    00000253'EF  D4  2E3A  9467          CLRL    DUMMY_PB                ; Assume there are no additional
                                     2E40  9468                                          ;  formative path blocks with
                                     2E40  9469                                          ;  associated system blocks.
                                     2E40  9470
                                     2E40  9471 ;
                                     2E40  9472 ; Store each system block in a singly linked list in SB buffer.
                                     2E40  9473 ;
                                     2E40  9474 ; REGISTER USAGE:
                                     2E40  9475 ;
                                     2E40  9476 ;       R3  -   Pointer into SB buffer where the system block will be stored.
                                     2E40  9477 ;       R9  -   Used to save a copy of R3 to be used after R3 is updated to
                                     2E40  9478 ;               point to the next free slot in SB buffer.
                                     2E40  9479 ;       R10 -   Pointer to system block in the SCS database.
                                     2E40  9480 ;
                                     2E40  9481
                53  00000017'EF  D0  2E40  9482 SB_SNAP:MOVL    SB_BUF_PNTR,R3          ; Get address of system block buffer.
                5A  00000000'GF  D0  2E47  9483          MOVL    G^SCS$GQ_CONFIG,R10     ; Get address of first system block.
                     18 AA   2F  E1  2E4E  9484 10$:     BBC     #47,SB$B_SYSTEMID(R10),-; Branch if not illegal type or system
                            34       2E52  9485                  SB_FIRST_LOOP           ;  block for UNA.
```

```
                5A   6A   D0  2E53  9486          MOVL    (R10),R10               ; Get address of next system block.
     00000000'8F   5A   D1  2E56  9487          CMPL    R10,#SCS$GQ_CONFIG      ; Any more system blocks?
                        2E5D  9488          JEQL    SB_LOOP_END             ; Branch if not.
                   EA   11  2E62  9489          BRB     10$                     ; Otherwise, loop for all system blocks.
                        2E64  9490 ;
                        2E64  9491 ; The following is skipped on the first pass because the first system block has
                        2E64  9492 ; no previous system block to link to.
                        2E64  9493 ;
                        2E64  9494
          00000029'EF   B7  2E64  9495 SB_LOOP:DECW    SB_DSCR+2               ; Anymore room left in the buffer?
                        2E6A  9496          JEQLU   SB_LOOP_END             ; Branch if not.
                5A   6A   D0  2E6F  9497 10$:    MOVL    (R10),R10               ; Get address of next system block.
     00000000'8F   5A   D1  2E72  9498          CMPL    R10,#SCS$GQ_CONFIG      ; Any more system blocks?
                        2E79  9499          JEQL    SB_LOOP_END             ; Branch if not.
             18 AA   2F   E0  2E7E  9500          BBS     #47,SB$B_SYSTEMID(R10),-; Ignore illegal and UNA system blocks.
                   EC        2E82  9501                  10$                     ;
             A0 A3   53   D0  2E83  9502          MOVL    R3,-SB$C_LENGTH(R3)     ; Link previous system block to this
                        2E87  9503                                              ;  one.
                        2E87  9504 SB_FIRST_LOOP:
                59   53   D0  2E87  9505          MOVL    R3,R9                   ; Save address where this SB will be
                        2E8A  9506                                              ;  stored.
     63   6A   0060 8F   28  2E8A  9507          MOVC3   #SB$C_LENGTH,(R10),(R3) ; Store system block in SB buffer.
                A0 A3   D4  2E90  9508          CLRL    -SB$C_LENGTH(R3)        ; Assume no more system blocks.
                        2E93  9509 ;
                        2E93  9510 ; Save the address of the local system block.  This will be used by the
                        2E93  9511 ; LISTENER_SNAP routine to link a dummy path block to the local system block
                        2E93  9512 ; if any listeners are found.
                        2E93  9513 ;
                        2E93  9514
          00000013'EF   D5  2E93  9515          TSTL    LOC_SB_ADRS             ; Have we already saved the address of
                        2E99  9516                                              ;  the local system block?
                   1B   12  2E99  9517          BNEQ    CSB_SNAP                ; Branch if we have.
             18 A9      D1  2E9B  9518          CMPL    SB$B_SYSTEMID(R9),-     ; Is this system block for the local
          00000005'EF       2E9E  9519                  LOCSYSID                ; system?
                   11   12  2EA3  9520          BNEQ    CSB_SNAP                ; Branch if not.
             1C A9      B1  2EA5  9521          CMPW    SB$B_SYSTEMID+4(R9),-   ; Must check all six bytes of ID.
          00000009'EF       2EA8  9522                  LOCSYSID+4              ;
                   07   12  2EAD  9523          BNEQ    CSB_SNAP                ; Branch if no match.
     00000013'EF   59   D0  2EAF  9524          MOVL    R9,LOC_SB_ADRS          ; Save address of local system block.
                        2EB6  9525
                        2EB6  9526
                        2EB6  9527 ;
                        2EB6  9528 ; This code executes if the display of Cluster System Blocks is enabled.
                        2EB6  9529 ; If there is a CSB associated with the last System Block stored in the SB
                        2EB6  9530 ; buffer, then it stores the CSB in the CBS buffer and adjusts SB$L_CSB to
                        2EB6  9531 ; point to the CSB.
                        2EB6  9532 ;
                        2EB6  9533 ; INPUTS:
                        2EB6  9534 ;       R3,R9  - Point to SB buffer where last SB was stored.
                        2EB6  9535 ;
                        2EB6  9536 ; REGISTER USAGE:
                        2EB6  9537 ;
                        2EB6  9538 ;       R3     - Used to point to CSB buffer where a CSB will be stored.
                        2EB6  9539 ;       R9     - Pointer to last SB stored the SB_BUFFER.
                        2EB6  9540 ;
                        2EB6  9541
                        2EB6  9542 CSB_SNAP:
```

```
                 02   E1  2EB6  9543          BBC     #CLS$V_CSB,-                ; Skip saving of CSB if MEMBERS class
     26 00000001'EF       2EB8  9544                  CLASS_SELECT,PB_SNAP        ;  is disabled.
           0408 8F   BB   2EBE  9545          PUSHR   #^M<R3,R10>
        5A   5C A9   D0   2EC2  9546          MOVL    SB$L_CSB(R9),R10            ; Get the address of the Cluster System
                          2EC6  9547                                              ;  Block for this system.
                 18   13  2EC6  9548          BEQL    CSB_SNAP_END                ; Branch if no CSB.
  53   0000001B'EF   D0   2EC8  9549          MOVL    CSB_BUF_PNTR,R3             ; Get the address to store the CSB in
                          2ECF  9550                                              ;  the CSB buffer.
        5C A9   53   D0   2ECF  9551          MOVL    R3,SB$L_CSB(R9)             ; Save pointer to CSB in SB.
     6A   00AC 8F   28   2ED3  9552          MOVC3   #CSB$C_LENGTH,(R10),-       ; Store the CSB in the CSB buffer.
                 63       2ED8  9553          (R3)                                ;
0000001B'EF   53   D0   2ED9  9554          MOVL    R3,CSB_BUF_PNTR             ; Save address to store next CSB.
                        2EE0  9555
                        2EE0  9556 CSB_SNAP_END:
           0408 8F   BA   2EE0  9557          POPR    #^M<R3,R10>
                        2EE4  9558
                        2EE4  9559 ;
                        2EE4  9560 ; Store the path blocks in singly linked lists in PB buffer. Link each list to
                        2EE4  9561 ; the associated system block by saving the address of the first path block
                        2EE4  9562 ; at offset SB$L_PBFL in the system block.
                        2EE4  9563 ;
                        2EE4  9564 ; Note: This loop does not store formative path blocks. See FORM_PB_SNAP:.
                        2EE4  9565 ;
                        2EE4  9566 ; INPUTS:
                        2EE4  9567 ;
                        2EE4  9568 ;       R9  -   Address of system block SB buffer that is associated with this
                        2EE4  9569 ;               path block.  (Passed from SB_SNAP routine)
                        2EE4  9570 ;       R10 -   Address of system block in SCS database for the last system
                        2EE4  9571 ;               block stored in SB buffer.
                        2EE4  9572 ;
                        2EE4  9573 ; REGISTER USAGE:
                        2EE4  9574 ;
                        2EE4  9575 ;       R3  -   Pointer into PB buffer where the path block will be stored.
                        2EE4  9576 ;       R7  -   Address of link to first path block on the system block.
                        2EE4  9577 ;       R9  -   Used to save a copy of R3 to be used after R3 is updated to
                        2EE4  9578 ;               point to the next free slot in PB buffer.
                        2EE4  9579 ;       R10 -   Pointer to path block in the SCS database.
                        2EE4  9580 ;
                        2EE4  9581
           0688 8F   BB   2EE4  9582 PB_SNAP:PUSHR   #^M<R3,R7,R9,R10>
              0C A9   D4   2EE8  9583          CLRL    SB$L_PBFL(R9)               ; Assume no paths on this system block.
  53   0000001F'EF   D0   2EEB  9584          MOVL    PB_BUF_PNTR,R3              ; Get address of first free slot in
                          2EF2  9585                                              ; path block buffer.
        57   0C AA   DE   2EF2  9586          MOVAL   SB$L_PBFL(R10),R7           ; Get address of forward link to first
                          2EF6  9587                                              ;  path block?
        57   0C AA   D1   2EF6  9588          CMPL    SB$L_PBFL(R10),R7           ; Are there any paths on this system
                          2EFA  9589                                              ;  block?
                        2EFA  9590          JEQL    PB_LOOP_END                 ; If not, exit this loop.
        0C A9   53   D0   2EFF  9591          MOVL    R3,SB$L_PBFL(R9)            ; Save address of first path block in
                          2F03  9592                                              ;  associated system block.
           5A   57   D0   2F03  9593          MOVL    R7,R10                      ; Get a copy of link address in R10.
           5A   6A   D0   2F06  9594          MOVL    (R10),R10                   ; Get forward link to the first path
                          2F09  9595                                              ;  block.
                 1A   11   2F09  9596          BRB     PB_FIRST_LOOP               ; Go save the first path block in
                          2F0B  9597                                              ;  PB buffer.
                          2F0B  9598
                          2F0B  9599 PB_LOOP:
```

```
                00000039'EF   B7  2F0B  9600          DECW    PB_DSCR+2               ; Any room left in the path block
                                  2F11  9601                                          ;  buffer?
                                  2F11  9602          JEQLU   PB_LOOP_END             ; Branch if not.
                     5A   6A   D0  2F16  9603          MOVL    (R10),R10               ; Get forward link to the (next) path
                                  2F19  9604                                          ;  block.
                     57   5A   D1  2F19  9605          CMPL    R10,R7                  ; Does the forward link point back to
                                  2F1C  9606                                          ;  first forward link?
                                  2F1C  9607          JEQL    PB_LOOP_END             ; If so, no more path blocks.
                   AC A3   53   D0  2F21  9608          MOVL    R3,-PB$C_LENGTH(R3)     ; Link previous path block to this one.
                                  2F25  9609
                                  2F25  9610 PB_FIRST_LOOP:
                     59   53   D0  2F25  9611          MOVL    R3,R9                   ; Save address where this path block
                                  2F28  9612                                          ;  will be stored.
           63   6A   0054 8F   28  2F28  9613          MOVC3   #PB$C_LENGTH,(R10),(R3) ; Copy path block to PB buffer.
                     AC A3   D4  2F2E  9614          CLRL    -PB$C_LENGTH(R3)        ; Assume no more path blocks.
                     E0 A3   D4  2F31  9615          CLRL    -PB$C_LENGTH+PB$L_CDTLST(R3)
                                  2F34  9616                                          ; Assume no CDT's on this path block.
                     B0 A3   D4  2F34  9617          CLRL    -PB$C_LENGTH+PB$SHC$L_NUM_CDT(R3)
                                  2F37  9618 ;
                                  2F37  9619 ; Get the number of SCS waiters on each path block by counting the entries
                                  2F37  9620 ; in the queue at offset PB$L_WAITQFL.  Save the count at the same offset in
                                  2F37  9621 ; place of the forward link (since it won't be used again)
                                  2F37  9622 ;
                                  2F37  9623 ; Fix the port name to be PA%0 rather than PA%"null".
                                  2F37  9624 ;
                                  2F37  9625 ; INPUTS:
                                  2F37  9626 ;
                                  2F37  9627 ;       R9  -   Address of path block that was last stored in PB buffer.
                                  2F37  9628 ;               (Passed from PB_SNAP routine)
                                  2F37  9629 ;       R10 -   Address of path block in the SCS database for the last path
                                  2F37  9630 ;               block stored in PB buffer.
                                  2F37  9631 ;
                                  2F37  9632 ; REGISTER USAGE:
                                  2F37  9633 ;
                                  2F37  9634 ;       R7  -   Address of link to first entry in SCS wait queue.
                                  2F37  9635 ;       R10 -   Pointer to entry in SCS wait queue.
                                  2F37  9636 ;
                                  2F37  9637
                                  2F37  9638 SCS_WAITER_SNAP:
                0480 8F   BB      2F37  9639          PUSHR   #^M<R7,R10>
           57     38 AA   DE      2F3B  9640          MOVAL   PB$L_WAITQFL(R10),R7    ; Get the address of the forward link
                                  2F3F  9641                                          ;  for the first entry in the SCS
                                  2F3F  9642                                          ;  wait queue.
                     5A   57   D0  2F3F  9643          MOVL    R7,R10                  ; Get a copy in R10.
                                  2F42  9644
                                  2F42  9645 SCS_WAITER_LOOP:
                     5A   6A   D0  2F42  9646          MOVL    (R10),R10               ; Get forward link to next entry in the
                                  2F45  9647                                          ;  SCS wait queue.
                     57   6A   D1  2F45  9648          CMPL    (R10),R7                ; Does the forward link point back to
                                  2F48  9649                                          ;  the first forward link?
                          04   13  2F48  9650          BEQL    SCS_WT_LP_END           ; If so, no more SCS waiters.
                          50   D6  2F4A  9651          INCL    R0                      ; Increase count of SCS waiters.
                          F4   11  2F4C  9652          BRB     SCS_WAITER_LOOP         ; Loop for all SCS waiters.
                                  2F4E  9653
                                  2F4E  9654 SCS_WT_LP_END:
           38 A9   50   D0        2F4E  9655          MOVL    R0,PB$L_WAITQFL(R9)     ; Store the number of SCS waiter at
                                  2F52  9656                                          ;  offset PB$L_WAITQFL in the PB.
```

```
                30  90  2F52  9657          MOVB    #^A/0/,-                    ; Fix the unit number in the port name
              27 A9         2F54  9658                  PB$T_LPORT_NAME+3(R9)    ;  to be "0" instead of NULL.
                          2F56  9659
          0480 8F   BA    2F56  9660          POPR    #^M<R7,R10>
                          2F5A  9661
                          2F5A  9662
                          2F5A  9663 ;
                          2F5A  9664 ; This routine is used to count the number of connections on a circuit and if
                          2F5A  9665 ; the display of connections is enabled, to store the connection descriptors in
                          2F5A  9666 ; CDT buffer.  All the CDT's for connections established over a particular path
                          2F5A  9667 ; are linked together and the first is linked to the associated path block.
                          2F5A  9668 ; The message associated with the CDT are also stored in CDT buffer immediately
                          2F5A  9669 ; following the CDT.  The number of connections on the circuit is stored in
                          2F5A  9670 ; the path block for the associated circuit at offset PB$SHC$L_NUM_PB.
                          2F5A  9671 ;
                          2F5A  9672 ; NOTE: This code does not check for connections that are in the LISTEN state.
                          2F5A  9673 ;       See LISTEN_CDT_SNAP:.
                          2F5A  9674 ;
                          2F5A  9675 ; INPUTS:
                          2F5A  9676 ;
                          2F5A  9677 ;       R9  -   Address of path block last stored in PB buffer.
                          2F5A  9678 ;       R10 -   Address of path block in SCS database for the last path block
                          2F5A  9679 ;               stored in PB buffer.
                          2F5A  9680 ;
                          2F5A  9681 ; REGISTER USAGE:
                          2F5A  9682 ;
                          2F5A  9683 ;       R3  -   Pointer into CDT buffer where the CDT will be stored.
                          2F5A  9684 ;       R4  -   Counter for copying or zeroing message.
                          2F5A  9685 ;       R5  -   Pointer to message when copying message into CDT buffer.
                          2F5A  9686 ;       R8  -   Used to count the number of connections on a given path.
                          2F5A  9687 ;       R10 -   Pointer to CDT in the SCS database.
                          2F5A  9688 ;
                          2F5A  9689
                           F5A  9690 ASSUME  SCS$T_SRC_PROC  EQ SCS$T_DST_PROC+16
                          2F5A  9691 ASSUME  SCS$B_CON_DAT   EQ SCS$T_SRC_PROC+16
                          2F5A  9692
                          2F5A  9693 CDT_SNAP:
              0408 8F     BB  2F5A  9694          PUSHR   #^M<R3,R10>
                  58      D4  2F5E  9695          CLRL    R8                      ; Clear counter for number of
                          2F60  9696                                              ;  connections.
      53  00000023'EF     D0  2F60  9697          MOVL    CDT_BUF_PNTR,R3         ; Get address of first free slot in
                          2F67  9698                                              ;  CDT buffer.
           5A   34 AA     DE  2F67  9699          MOVAL   PB$L_CDTLST(R10),R10    ; Get the address of the first forward
                          2F6B  9700                                              ;  link for the CDT's associated with
                          2F6B  9701                                              ;  this path block.
           5A   6A     D0  2F6B  9702          MOVL    (R10),R10               ; Are there any CDTs on this path
                          2F6E  9703                                              ;  block?
                          2F6E  9704 5$:      JEQL    CDT_SNAP_DONE           ; If not go on to next path block.
                  58      D6  2F73  9705          INCL    R8                      ; Increase count of connections on this
                          2F75  9706                                              ;  circuit.
          00000060 8F     D3  2F75  9707          BITL    #CLS$M_CNT!CLS$M_CRD,-  ; Is the display of counters or credits
          00000001'EF         2F7B  9708                  CLASS_SELECT            ;  enabled?
                  20      12  2F80  9709          BNEQ    20$                     ; Branch if so.
                  04      E1  2F82  9710          BBC     #CLS$V_CON,-            ; If the display of connections is also
    1E 00000001'EF              2F84  9711                  CLASS_SELECT,30$        ;  disabled, then skip storing of CDT.
                 28 AA    B1  2F8A  9712          CMPW    CDT$W_STATE(R10),-      ; Is the connection in the open state?
                  02          2F8D  9713                  #CDT$C_OPEN             ;
```

```
                     0A   12  2F8E  9714          BNEQU   10$                     ; Branch if not.
                     0B   E0  2F90  9715          BBS     #CLS$V_OPEN_CON,-       ; Branch to link first CDT if we're
          0A 00000001'EF       2F92  9716                  CLASS_SELECT,20$        ;  displaying open connections.
                     0E   11  2F98  9717          BRB     30$                     ; Otherwise skip first link.
                     0C   E1  2F9A  9718 10$:     BBC     #CLS$V_NOOPEN_CON,-     ; Branch if we're not displaying
          06 00000001'EF       2F9C  9719                  CLASS_SELECT,30$        ;  connections in any other state.
            34 A9    53   D0  2FA2  9720 20$:     MOVL    R3,PB$L_CDTLST(R9)      ; Else, link the first CDT to the PB.
                     3D   11  2FA6  9721          BRB     CDT_FIRST_LOOP          ; Don't link to previous CDT.
                          2FA8  9722
             5A   6C AA   D0  2FA8  9723 30$:     MOVL    CDT$L_CDTLST(R10),R10   ; Get the forward link to the next CDT.
                     C0   11  2FAC  9724          BRB     5$                      ; And loop.
                          2FAE  9725
                          2FAE  9726 CDT_LOOP:
          00000041'EF    B7  2FAE  9727          DECW    CDT_DSCR+2              ; Any room left in connections buffer?
                     53   13  2FB4  9728          BEQLU   CDT_SNAP_DONE           ; Branch if not.
             5A   6C AA   D0  2FB6  9729          MOVL    CDT$L_CDTLST(R10),R10   ; Get forward link to CDT.
                     4D   13  2FBA  9730          BEQL    CDT_SNAP_DONE           ; Branch if there are no (more) CDT's.
                   28 AA  B1  2FBC  9731          CMPW    CDT$W_STATE(R10),-      ; Is the connection in the open state?
                     02         2FBF  9732                  #CDT$C_OPEN             ;
                     11   12  2FC0  9733          BNEQU   10$                     ; Branch if not.
                     58   D6  2FC2  9734          INCL    R8                      ; Increase count of connections on this
                          2FC4  9735                                              ;  circuit.
                     D3   2FC4  9736          BITL    #<CLS$M_OPEN_CON!-      ; Are we displaying open connections,
                          2FC5  9737                  CLS$M_CNT!CLS$M_CRD>,-  ;  counters or credits?
    00000001'EF    00000860 8F   2FC5  9738                  CLASS_SELECT            ;
                     0F   12  2FCF  9739          BNEQ    20$                     ; Branch if so.
                     DB   11  2FD1  9740          BRB     CDT_LOOP                ; Otherwise, don't link CDT.
                     D3   2FD3  9741 10$:     BITL    #<CLS$M_NOOPEN_CON!-    ; Are we displaying connections that
                          2FD4  9742                  CLS$M_CRD!CLS$M_CRD>,-  ;  aren't open, counters, or credits?
    00000001'EF    00001040 8F   2FD4  9743                  CLASS_SELECT            ;
                     CE   13  2FDE  9744          BEQL    CDT_LOOP                ; If not, don't link CDT.
                FF54 C3   53   D0  2FE0  9745 20$:     MOVL    R3,-                    ; Link previous CDT to this one.
                          2FE5  9746                  -<SHC$C_CDT_LENGTH+32>(R3)  ; Total length of CDT is number of
                          2FE5  9747                                              ;  bytes stored + 32 byte message.
                          2FE5  9748 CDT_FIRST_LOOP:
                     83   D4  2FE5  9749          CLRL    (R3)+                   ; Assume this is the last CDT;
                          2FE7  9750                                              ;  Zero forward link.
                   0088 8F   28  2FE7  9751          MOVC3   #SHC$C_CDT_LENGTH-4,-   ; Save a portion of the CDT in CDT
                     14 AA       2FEB  9752                  SHC$L_CDT_BEGIN(R10),-  ;  buffer.  (Length -4 because forward
                     63          2FED  9753                  (R3)                    ;  link is not copied.)
                          2FEE  9754                                              ;  Note: R4 is cleared.
             55   2C A0   D0  2FEE  9755          MOVL    CDT$L_SCSMSG(R10),R5    ; Is there a message associated with
                          2FF2  9756                                              ;  this CDT?
                     0D   13  2FF2  9757          BEQL    NO_MESSAGE              ; Branch if not.
             55   04 A5   DE  2FF4  9758          MOVAL   SCS$T_DST_PROC(R5),R5   ; Get the address of the process names
                          2FF8  9759                                              ;  in the message.
                83   85   7D  2FF8  9760 MESSAGE:MOVQ    (R5)+,(R3)+             ; Move message 8 bytes at a time.
                F9 54   04   F2  2FFB  9761          AOBLSS  #4,R4,MESSAGE           ; Loop for 32 bytes.
                     AD   11  2FFF  9762          BRB     CDT_LOOP                ; Loop for all CDT's.
                          3001  9763
                          3001  9764 NO_MESSAGE:
                     83   7C  3001  9765          CLRQ    (R3)+                   ; Zero out the messages area.
                FA 54   04   F2  3003  9766          AOBLSS  #4,R4,NO_MESSAGE        ; Loop for 32 bytes.
                     A5   11  3007  9767          BRB     CDT_LOOP                ; Loop for all CDT's.
                          3009  9768
                          3009  9769 CDT_SNAP_DONE:
             04 A9   58   D0  3009  9770          MOVL    R8,PB$SHC$L_NUM_CDT(R9) ; Save number of connections on this
```

```
                     300D  9771                                   ;  path in the path block.
00000023'EF   53   D0 300D  9772            MOVL    R3,CDT_BUF_PNTR         ; Save address of next free slot in
                     3014  9773                                   ;  CDT buffer.
          0408 8F   BA 3014  9774            POPR    #^M<R3,R10>
                     3018  9775
                     3018  9776 CDT_SNAP_END:
             FEF0    31 3018  9777            BRW     PB_LOOP                 ; Loop for all path blocks on this
                     301B  9778                                   ;  system block.
                     301B  9779
                     301B  9780 PB_LOOP_END:
0000001F'EF   53   D0 301B  9781            MOVL    R3,PB_BUF_PNTR          ; Save address of next free entry in
                     3022  9782                                   ;  PB buffer.
          0688 8F   BA 3022  9783            POPR    #^M<R3,R7,R9,R10>
                     3026  9784
             FE3B    31 3026  9785            BRW     SB_LOOP                 ; Loop for all system blocks.
                     3029  9786
                     3029  9787 SB_LOOP_END:
                     3029  9788
00000017'EF   53   D0 3029  9789            MOVL    R3,SB_BUF_PNTR          ; Save address of next free entry in
                     3030  9790                                   ;  SB buffer.
                     3030  9791
                     3030  9792 ;
                     3030  9793 ; This routine executes if the display of circuits in states other than open
                     3030  9794 ; is enabled.  It scans the SCS directory and checks for local processes with
                     3030  9795 ; connections in the LISTEN state.  If any listening CDT's are found, they are
                     3030  9796 ; copied into CDT buffer and linked together.  The first CDT is linked to a
                     3030  9797 ; dummy path block which is in turn linked to the system block for the local
                     3030  9798 ; system.  The local process name is copied from the SCS directory entry and
                     3030  9799 ; stored with the CDT.  The remote process name is zero out.
                     3030  9800 ;
                     3030  9801 ; In the event that there is no system block for the local system block, the
                     3030  9802 ; dummy path block is linked to the dummy system block.
                     3030  9803 ;
                     3030  9804 ; REGISTER USAGE:
                     3030  9805 ;
                     3030  9806 ;       R3  -   Pointer into CDT buffer where the CDT will be stored.
                     3030  9807 ;       R4  -   Used to determine the address where the link to the first
                     3030  9808 ;               listening CDT should be stored.  This address will either be
                     3030  9809 ;               in the dummy path block or in the last path block on the local
                     3030  9810 ;               system system block.
                     3030  9811 ;       R5  -   Used to copy the local process name from the directory entry to
                     3030  9812 ;               CDT buffer.
                     3030  9813 ;       R6  -   Pointer to SCS connection descriptor list.
                     3030  9814 ;       R7  -   Pointer to entry in SCS directory.
                     3030  9815 ;       R9  -   Pointer to dummy path block.
                     3030  9816 ;       R10 -   Used to convert index from the SCS directory to the address of
                     3030  9817 ;               the CDT.
                     3030  9818 ;
                     3030  9819
                     3030  9820 LISTENER_SNAP:
              0C   E0 3030  9821            BBS     #CLS$V_NOOPEN_CON,-     ; Execute this code if display of
  03 00000001'EF       3032  9822                    CLASS_SELECT,5$         ;  connections in states other than
                     3038  9823                                   ;  open is enabled.
            0099   31 3038  9824            BRW     COMMON_SNAP             ; Other wise jump to common code.
53  00000023'EF   D0 303B  9825 5$:         MOVL    CDT_BUF_PNTR,R3         ; Get address to store next CDT in
                     3042  9826                                   ;  CDT buffer.
59  00000253'EF   DE 3042  9827            MOVAL   DUMMY_PB,R9             ; Get address of dummy path block.
```

```
                34 A9   D4  3049  9828          CLRL    PB$L_CDTLST(R9)         ; Assume there are no listeners.
57  00000000'GF     DE  304C  9829          MOVAL   G^SCS$GQ_DIRECT,R7       ; Get address of first SCS directory
                        3053  9830                                              ;  forward link.
56  00000000'GF     D0  3053  9831          MOVL    G^SCS$GL_CDL,R6         ; Get the address of the SCS connection
                        305A  9832                                              ;  descriptor list.
                        305A  9833
                        305A  9834  SCS_DIR_LOOP:
        00000041'EF     B5  305A  9835          TSTW    CDT_DSCR+2              ; Is there any room on the CDT buffer?
                    6B  13  3060  9836          BEQLU   LISTENER_SNAP_END       ; Branch if not.
            57      67  D0  3062  9837          MOVL    (R7),R7                 ; Get the forward link to first SCS
                        3065  9838                                              ;  directory entry.
00000000'8F     57      D1  3065  9839          CMPL    R7,#SCS$GQ_DIRECT       ; Does the link point back to the
                        306C  9840                                              ;  address of the first link?
                    5F  13  306C  9841          BEQL    LISTENER_SNAP_END       ; If so, no more directory entries.
        5A  2C A7       3C  306E  9842          MOVZWL  SDIR$L_CONID(R7),R10    ; Get index value for this CDT.
         5A   664A      D0  3072  9843          MOVL    (R6)[RTO],R10           ; Get address of the CDT.
                28 AA   B1  3076  9844          CMPW    CDT$W_STATE(R10),-      ; Is this CDT in the "listen" state?
                    01      3079  9845                  #CDT$C_LISTEN           ;
                    DE  12  307A  9846          BNEQ    SCS_DIR_LOOP            ; If not, look at next directory entry.
                34 A9   D5  307C  9847          TSTL    PB$L_CDTLST(R9)         ; Have we already linked the first
                        307F  9848                                              ;  listening CDT to the dummy PB?
                    28  12  307F  9849          BNEQ    30$                     ; Branch if we have.
        34 A9       53  D0  3081  9850          MOVL    R3,PB$L_CDTLST(R9)      ; Otherwise, link it now.
54  00000013'EF     D0  3085  9851          MOVL    LOC_SB_ADRS,R4          ; Get the address of the system block
                        308C  9852                                              ;  for the local system.
                    07  12  308C  9853          BNEQ    5$                      ; Branch if there is a system block for
                        308E  9854                                              ;  the local system.
54  000001F3'EF     DE  308E  9855          MOVAL   DUMMY_SB,R4             ; Otherwise, get the address of the
                        3095  9856                                              ;  dummy system block.
    54  0C A4      DE  3095  9857  5$:      MOVAL   SB$L_PBFL(R4),R4        ; Get the address of the link to the
                        3099  9858                                              ;  first path block on this system
                        3099  9859                                              ;  block.
                    64  D5  3099  9860          TSTL    (R4)                    ; Are there any path blocks on this
                        309B  9861                                              ;  system block?
                    07  13  309B  9862          BEQL    20$                     ; If not, link the dummy path block
                        309D  9863                                              ;  directly to the system block.
        54          64  D0  309D  9864  10$:     MOVL    (R4),R4                 ; Get the link to the next path block.
                    64  D5  30A0  9865          TSTL    (R4)                    ; Is this the last path block?
                    F9  12  30A2  9866          BNEQ    10$                     ; Loop of not.
        64          59  D0  30A4  9867  20$:     MOVL    R9,(R4)                 ; Link the dummy PB to the last path
                        30A7  9868                                              ;  block on this system block.
                    05  11  30A7  9869          BRB     40$                     ; Don't link listening CDT to any
                        30A9  9870                                              ;  previous CDT's.
    FF54 C3         53  D0  30A9  9871  30$:     MOVL    R3,-                    ; Link previous CDT to this one.
                        30AE  9872              -<SHC$C_CDT_LENGTH+32>(R3)      ; Total length of CDT is number of
                        30AE  9873                                              ;  bytes stored + 32 byte message.
                    83  D4  30AE  9874  40$:     CLRL    (R3)+                   ; Assume this is the last CDT in the
                        30B0  9875                                              ;  listen state; Zero forward link.
        0088 8F     28  30B0  9876          MOVC3   #SHC$C_CDT_LENGTH-4,-   ; Save a portion of the CDT in CDT
              14 AA         30B4  9877                  SHC$L_CDT_BEGIN(R10),-  ;  buffer.  (Length -4 because forward
                    63      30B6  9878                  (R3)                    ;  link is not copied.)
                        30B7  9879                                              ;  Note: R4 is cleared.
    55  0C A7      DE  30B7  9880          MOVAL   SDIR$B_PROCNAM(R7),R5   ; Get address of local process name in
                        30BB  9881                                              ;  directory entry.
        83      85  7D  30BB  9882          MOVQ    (R5)+,(R3)+             ; Save local process name immediately
        83      85  7D  30BE  9883          MOVQ    (R5)+,(R3)+             ;  following the CDT.
                    83  7C  30C1  9884          CLRQ    (R3)+                   ; Zero out remote process name.
```

```
                     83   7C  30C3  9885          CLRQ    (R3)+                   ; All 16 bytes...
           00000041'EF    B7  30C5  9886          DECW    CDT_DSCR+2              ; Count down the number of entries that
                         30CB  9887                                               ;  will fit in the CDT buffer.
                     8D  11  30CB  9888          BRB     SCS_DIR_LOOP            ; Loop for all directory entries.
                         30CD  9889
                         30CD  9890 LISTENER_SNAP_END:
       00000023'EF   53   D0  30CD  9891          MOVL    R3,CDT_BUF_PNTR         ; Save address to store next CDT.
                         30D4  9892
                         30D4  9893 COMMON_SNAP:
                     0A  E0  30D4  9894          BBS     #CLS$V_NOOPEN_CIR,-     ; Continue with the following code
           00000001'EF       30D6  9895                  CLASS_SELECT,-          ;  if we're displaying formative paths.
                     03      30DB  9896                  LOCAL_SNAP              ;
                   01F2  31  30DC  9897          BRW     SNAPSHOT_END            ; Otherwise finish up.
                         30DF  9898
                         30DF  9899 ;
                         30DF  9900 ; This code is always executed for the local report.  It is also executed for
                         30DF  9901 ; the cluster report if circuits in states other than open are being displayed.
                         30DF  9902 ; It scans the device list looking for PAA0 or PAB0 to determine what ports are
                         30DF  9903 ; on the local system so that the associated DDBs, UCBs, and PDTs can be
                         30DF  9904 ; located.
                         30DF  9905 ;
                         30DF  9906 ; REGISTER USAGE:
                         30DF  9907 ;
                         30DF  9908 ;       R3  -   Address in PDT_BUF where the PDT will be stored.
                         30DF  9909 ;       R8  -   Pointer into MISC_BUF where miscellaneous information for the
                         30DF  9910 ;               port will be stored.  Data for PAA0 starts at MISC_BUF.  Data
                         30DF  9911 ;               for PAB0 starts at MISC_BUF+MISC_BUF_LEN.
                         30DF  9912 ;       R10 -   Used as pointer to DDB's.
                         30DF  9913 ;
                         30DF  9914
                         30DF  9915 LOCAL_SNAP:
   53   000002A7'EF      DE  30DF  9916          MOVAL   PDT_BUF,R3              ; Get address of PDT buffer.
   5A   00000000'GF      D0  30E6  9917          MOVL    G^IOC$GL_DEVLIST,R10    ; Get address of device list.
   58   00000419'EF      DE  30ED  9918          MOVAL   MISC_BUF-MISC_BUF_LEN,R8; Initialize pointer to MISC_BUF.
                         30F4  9919
                         30F4  9920 DDB_LOOP:
             5A   6A      D0  30F4  9921          MOVL    (R10),R10               ; Get address of DDB.
                  03      12  30F7  9922          BNEQ    10$                     ; Continue if more DDB's.
                01D5      31  30F9  9923          BRW     SNAPSHOT_END            ; Otherwise, done if no more DDBs.
41415003 8F    14   AA      D1  30FC  9924 10$:     CMPL    DDB$T_NAME(R10),#PAAC   ; Is this DDB for PAA0?
                  0A      13  3104  9925          BEQL    20$                     ; Branch if so.
42415003 8F    14   AA      D1  3106  9926          CMPL    DDB$T_NAME(R10),#PABC   ; Is this DDB for PAB0?
                  E4      12  310E  9927          BNEQ    DDB_LOOP                ; If not continue looking.
             58   2E      C0  3110  9928 20$:     ADDL    #MISC_BUF_LEN,R8        ; Get address to store data for PA%0 in
                         3113  9929                                               ;  miscellaneous buffer.
                  00      E0  3113  9930          BBS     #RPT$V_CLSTR,-          ; Branch to the formative path block
           00000000'EF       3115  9931                  CURRENT_REPORT,-         ; snapshot routine if we are displaying
                  03       311A  9932                  FORM_PB_SNAP            ; the cluster report.
                00BB      31  311B  9933          BRW     PORT_SNAP               ; Otherwise, branch to the port
                         311E  9934                                               ;  snapshot routine.
                         311E  9935
                         311E  9936 ;
                         311E  9937 ; This code executes only if the cluster report is selected and display of
                         311E  9938 ; circuits in states other than open is enabled.  This routine stores formative
                         311E  9939 ; path blocks in PB buffer.  It checks for a formative system block associated
                         311E  9940 ; with the formative path block.
                         311E  9941 ;
```

```
                        ;  If there is no formative system block associated with the formative path
                        ;  block then it links the formative path block to a dummy system block.
                        ;
                        ;  If the formative path block does have a formative system block associated it
                        ;  checks if the system ID of the formative system block matchs that of any of
                        ;  the system blocks stored in SB buffer.  If there is no match, then the
                        ;  formative system block is stored in SB buffer and the formative path block is
                        ;  linked to it.
                        ;
                        ;  If there is a match, then the incarnation number of the formative system
                        ;  block is checked against the incarnation number of the system block stored in
                        ;  SB buffer.  If the incarnation numbers are the same, then the formative path
                        ;  block is linked to the system block in SB buffer.  If the incarnation numbers
                        ;  are different, then the formative system block is stored in SB buffer and the
                        ;  formative path block is linked to it.
                        ;
                        ;  INPUTS:
                        ;
                        ;       R10 -   Points to base address of DDB.
                        ;
                        ;  REGISTER USAGE:
                        ;
                        ;       R3  -   Used to scan the system blocks in SB buffer for a system whose
                        ;               ID matches the formative system block.  If found points to
                        ;               the system block.  If not found points to dummy system block.
                        ;           -   Used to find the address last formative path block on the
                        ;               formative system block or the dummy system block.
                        ;           -   Used as pointer into PB buffer where the formative path block
                        ;               is to be stored.
                        ;       R7  -   Address of forward link to first formative path block on the
                        ;               PDT.
                        ;       R9  -   Address of formative system block if available.
                        ;       R10 -   Used to get address of UCB from DDB and address of PDT from
                        ;               UCB.
                        ;           -   Pointer to formative path block.
                        ;
                        ;
                        ;
                        FORM_PB_SNAP:
                        0A E1   BBC     #CLS$V_NOOPEN_CIR,-     ; If display of circuits in states
CE 00000001'EF                  CLASS_SELECT,DDB_LOOP   ;  other than open is disabled then
                                                        ;  continue scanning for DDB's.
070C 8F BB              PUSHR   #^M<R2,R3,R8,R9,R10>
   5A 04 AA DO          MOVL    DDB$L_UCB(R10),R10      ; Get address of UCB0 for this port.
 5A 0084 CA DO          MOVL    UCB$L_PDT(R10),R10      ; Get address of PDT for this port.
                        JEQL    FORM_PB_DONE            ; Branch if no PDT.
 57 0174 CA DE          MOVAL   PDT$G_FORMPB(R10),R7    ; Get address of the forward link for
                                                        ;  the first formative path block on
                                                        ;  this PDT.
    5A 57 DO            MOVL    R7,R10                  ; Get copy in R10.
                        FORM_PB_LOOP:
    5A 6A DO            MOVL    (R10),R10               ; Get address of formative path block.
    57 5A D1            CMPL    R10,R7                  ; Does the forward link point to the
                                                        ;  address of the first link?
                        JEQL    FORM_PB_DONE            ; If so, no (more) formative path
                                                        ;  blocks.  Go check for another port.
 59 30 AA DO            MOVL    PB$L_SBLINK(R10),R9     ; Get formative system block link.
```

```
                4A   13  314F  9999          BEQL    40$                     ; Branch if there is no system block
                                3151 10000                                    ;  associated with this path block.
        58  00000047'EF   D0    3151 10001          MOVL    VAR_BUF_ADRS,R8         ; Point to the system block buffer.
              18 A8   D1    3158 10002 10$:     CMPL    SB$B_SYSTEMID(R8),-     ; Does the system ID of the formative
              18 A9               315B 10003          SB$B_SYSTEMID(R9)       ;  system block match the ID of any of
                                315D 10004                                    ;  the system blocks stored in SB
                                315D 10005                                    ;  buffer?
                 15   12    315D 10006          BNEQ    20$                     ; Branch if not.
              1C A8   B1    315F 10007          CMPW    SB$B_SYSTEMID+4(R8),-   ; Check all six bytes.
              1C A9               3162 10008          SB$B_SYSTEMID+4(R9)     ;
                 0E   12    3164 10009          BNEQ    20$                     ; Branch if no match.
              2C A8   D1    3166 10010          CMPL    SB$Q_SWINCARN(R8),-     ; Does the boot time in the formative
              2C A9               3169 10011          SB$Q_SWINCARN(R9)       ;   system block match the system block
                                316B 10012                                    ;   we've found?
                 0C   12    316B 10013          BNEQ    30$                     ; Branch if not.
              30 A8   D1    316D 10014          CMPL    SB$Q_SWINCARN+4(R8),-   ; Check the full boot time.
              30 A9               3170 10015          SB$Q_SWINCARN+4(R9)     ;
                 2E   13    3172 10016          BEQL    50$                     ; Branch if the boot time matches.
                                3174 10017
           58   68   D0    3174 10018 20$:     MOVL    SB$L_FLINK(R8),R8       ; Get address of next system block.
                 DF   12    3177 10019          BNEQ    10$                     ; Loop for all system blocks in SB
                                3179 10020                                    ;  buffer.
                                3179 10021
                                3179 10022 ;
                                3179 10023 ; Save the formative system block if it's ID doesn't match any of the system
                                3179 10024 ; blocks in SB buffer or if the ID matches, but the incarnation number doesn't.
                                3179 10025 ;
                                3179 10026
 53   00000017'EF   D0    3179 10027 30$:     MOVL    SB_BUF_PNTR,R3          ; Get the address in SB buffer to store
                                3180 10028                                    ;  system block.
       A0 A3   53   D0    3180 10029          MOVL    R3,-SB$C_LENGTH(R3)     ; Link previous system block to this
                                3184 10030                                    ;  one.
          58   53   D0    3184 10031          MOVL    R3,R8                   ; Save the address where this system
                                3187 10032                                    ;  block will be stored.
63   69   0060 8F   28    3187 10033          MOVC3   #SB$C_LENGTH,(R9),(R3)  ; Save the system block in SB buffer.
 00000017'EF   53   D0    318D 10034          MOVL    R3,SB_BUF_PNTR          ; Save address to store next system
                                3194 10035                                    ;  block.
                68   D4    3194 10036          CLRL    (R8)                    ; Indicate that this is the last system
                                3196 10037                                    ;  block.
             0C A8   D4    3196 10038          CLRL    SB$L_PBFL(R8)           ; Indicate that there are no path
                                3199 10039                                    ;  blocks on this system block.
                07   11    3199 10040          BRB     50$                     ; Jump to common code.
 58   000001F3'EF   DE    319B 10041 40$:     MOVAL   DUMMY_SB,R8             ; Get the address of the fake system
                                31A2 10042                                    ;  block.
       58   0C A8   DE    31A2 10043 50$:     MOVAL   SB$L_PBFL(R8),R8        ; Get address of first forward link to
                                31A6 10044                                    ;  the path blocks on this system
                                31A6 10045                                    ;  block.
                68   D5    31A6 10046 60$:     TSTL    (R8)                    ; Is this the last formative path
                                31A8 10047                                    ;  block?
                05   13    31A8 10048          BEQL    70$                     ; Branch if so.
             58   68   D0    31AA 10049          MOVL    (R8),R8                 ; Get link to next formative PB.
                 F7   11    31AD 10050          BRB     60$                     ; Loop till last FORM_PB is found.
 68   0000001F'EF   D0    31AF 10051 70$:     MOVL    PB_BUF_PNTR,(R8)        ; Link the new formative path block
                                31B6 10052                                    ;  to the last one.
          53   68   D0    31B6 10053          MOVL    (R8),R3                 ; Get address where the formative PB
                                31B9 10054                                    ;  will be stored.
63   6A   0054 8F   28    31B9 10055          MOVC3   #PB$C_LENGTH,(R10),(R3) ; Save the formative PB in PB buffer.
```

```
                AC A3   D4  31BF 10056          CLRL    -PB$C_LENGTH(R3)        ; Indicate that the formative PB is
                                31C2 10057                                      ;  now the last path block on this
                                31C2 10058                                      ;  system block.
                E0 A3   D4  31C2 10059          CLRL    -PB$C_LENGTH+PB$L_CDTLST(R3)
                                31C5 10060                                      ; Zero the link to any connections.
                B0 A3   D4  31C5 10061          CLRL    -PB$C_LENGTH+PB$SHC$L_NUM_CDT(R3)
                                31C8 10062                                      ; Zero the number of connections on the
                                31C8 10063                                      ;  formative path block.
0000001F'EF   53    D0  31C8 10064          MOVL    R3,PB_BUF_PNTR          ; Save the address to store the next
                                31CF 10065                                      ;  PB.
              FF6E  31  31CF 10066          BRW     FORM_PB_LOOP            ; Loop for all formative path blocks.
                                31D2 10067
                                31D2 10068 FORM_PB_DONE:
            070C 8F   BA  31D2 10069          POPR    #^M<R2,R3,R8,R9,R10>
              FF1B  31  31D6 10070          BRW     DDB_LOOP                ; Go check for another port.
                                31D9 10071
                                31D9 10072 ;
                                31D9 10073 ; This code executes only if the local port report in selected.  It stores
                                31D9 10074 ; miscellaneous data about the port and data from the interlocked queues in
                                31D9 10075 ; MISC_BUF.  It stores the PDT's in PDT_BUF.
                                31D9 10076 ;
                                31D9 10077 ; INPUTS:
                                31D9 10078 ;
                                31D9 10079 ;       R10 -   Points to base address of DDB.
                                31D9 10080 ;       R8  -   Pointer into MISC_BUF where miscellaneous information for the
                                31D9 10081 ;               port will be stored.
                                31D9 10082 ;
                                31D9 10083 ; REGISTER USAGE:
                                31D9 10084 ;
                                31D9 10085 ;       R7  -   Used to get address of UCB from DDB and address of PDT from
                                31D9 10086 ;               UCB.
                                31D9 10087 ;       R8  -   Pointer to MISC_BUF.
                                31D9 10088 ;       R10 -   Pointer to DDB.
                                31D9 10089 ;
                                31D9 10090
                                31D9 10091 PORT_SNAP:
                                31D9 10092
                                31D9 10093 ;
                                31D9 10094 ; Convert the port name for this port from "PA%" to "PA%0" and save in
                                31D9 10095 ; MISC_BUF.
                                31D9 10096 ;
  14 AA    F8 8F    78  31D9 10097          ASHL    #-8,DDB$T_NAME(R10),-   ; Get the port name shifted left one
                   68  31DE 10098                  MB$L_PORT_NAME(R8)      ;  byte.
    30000000 8F    C8  31DF 10099          BISL    #^X30000000,-           ; AND in a zero at the end of the name.
                   68  31E5 10100                  MB$L_PORT_NAME(R8)      ;
          14 A8    D4  31E6 10101          CLRL    MB$L_FORM_PB(R8)        ; Initialize count of formative path
                                31E9 10102                                      ;  blocks on this port. (In MISC_BUF)
     57   04 AA    D0  31E9 10103          MOVL    DDB$L_UCB(R10),R7       ; Get address of UCB0 for this port.
          41 A7    9A  31ED 10104          MOVZBL  UCB$B_DEVTYPE(R7),-     ; Get the device type for this port
          08 A8        31F0 10105                  MB$L_DEVTYPE(R8)        ;  from UCB0 and save in MISC_BUF.
          64 A7    3C  31F2 10106          MOVZWL  UCB$W_STS(R7),-         ; Get device status for this port
          0C A8        31F5 10107                  MB$L_STS(R8)            ;  from UCB0 and save in MISC_BUF.
        0080 C7    D0  31F7 10108          MOVL    UCB$B_ERTCNT(R7),-      ; Get current retry count, maximum
          10 A8        31FB 10109                  MB$B_ERTCNT(R8)         ;  retry count, and error count from
                                31FD 10110                                      ;  UCB0 and save in MISC_BUF.
     57   0084 C7    D0  31FD 10111          MOVL    UCB$L_PDT(R7),R7        ; Get address of PDT for this port.
                        3202 10112          JEQL    FREE_BUF_SNAP           ; Branch if no PDT.
```

```
                     3207 10113
                     3207 10114 ;
                     3207 10115 ; Count the number of formative circuits and save in MISC_BUF.
                     3207 10116 ; Save the port number in MISC_BUF.
                     3207 10117 ; Copy a portion of the PDT in PDT_BUF.
                     3207 10118 ; Count the number of pool waiters and store in MISC_BUF.
                     3207 10119 ;
                     3207 10120 ; INPUTS:
                     3207 10121 ;
                     3207 10122 ;       R3  -   Pointer into PDT_BUF where PDT is to be stored.
                     3207 10123 ;       R7  -   Base address of PDT.
                     3207 10124 ;       R8  -   Base address to store data for this port in MISC_BUF.
                     3207 10125 ;
                     3207 10126 ; REGISTER USAGE:
                     3207 10127 ;
                     3207 10128 ;       R5  -   Pointer to formative path blocks.
                     3207 10129 ;       R5  -   Pointer to entries in pool waiter queue.
                     3207 10130 ;       R6  -   Address of link to first formative path block on the PDT.
                     3207 10131 ;           -   Address of loopback data gram buffer.
                     3207 10132 ;           -   Pointer to first address in PDT to be saved in PDT_BUF.
                     3207 10133 ;           -   Address of link to first pool waiter queue entry.
                     3207 10134 ;       R8  -   Pointer to MISC_BUF.
                     3207 10135 ;
56   0174 C7   DE    3207 10136          MOVAL   PDT$Q_FORMPB(R7),R6     ; Get address of the forward link for
                     320C 10137                                          ;  the first formative path block on
                     320C 10138                                          ;  this PDT.
     55   56   D0    320C 10139          MOVL    R6,R5                   ; Get copy in R5.
                     320F 10140
                     320F 10141 PORT_FORM_PB_LOOP:
     55   65   D0    320F 10142          MOVL    (R5),R5                 ; Get address of formative path block.
     56   55   D1    3212 10143          CMPL    R5,R6                   ; Does the forward link point to the
                     3215 10144                                          ;  address of the first link?
          05   13    3215 10145          BEQL    PORT_FORM_PB_LOOP_END   ; If so, no (more) formative path
                     3217 10146                                          ;  blocks.  Go check for another port.
       14 A8   D6    3217 10147          INCL    MB$L_FORM_PB(R8)        ; Increase count of formative circuits.
          F3   11    321A 10148          BRB     PORT_FORM_PB_LOOP       ; Loop for all formative paths.
                     321C 10149
                     321C 10150 PORT_FORM_PB_LOOP_END:
56   0184 C7   D0    321C 10151          MOVL    PDT$L_LBDG(R7),R6       ; Get address of loopback datagram
                     3221 10152                                          ;  buffer.
       0C A6   9A    3221 10153          MOVZBL  PPD$B_PORT(R6),-        ; Save the port number of this port in
          04 A8                                  MB$L_PORT_NUM(R8)       ;  MISC_BUF.
56   00B4 C7   DE    3226 10154          MOVAL   SHC$L_PDT_BEGIN(R7),R6  ; Get address to start copying PDT.
     00D0 8F   28    322B 10155          MOVC3   #SHC$L_PDT_LENGTH,-     ; Get a portion of the PDT into
     63   66    322F 10156                  (R6),(R3)                   ;  PDT_BUF.
55   00AC C7   DE    3231 10157          MOVAL   PDT$L_WAITQFL(R7),R5    ; Get address of pool waiter queue
                     3236 10158                                          ;  forward link.
     56   55   D0    3236 10159          MOVL    R5,R6                   ; Save a copy in R5.
       18 A8   D4    3239 10160          CLRL    MB$L_PL_WAITERS(R8)     ; Initialize number of pool waiters on
                     323C 10161                                          ;  this port.
                     323C 10162
                     323C 10163
                     323C 10164 POOL_WAITER_LOOP:
     55   65   D0    323C 10165          MOVL    (R5),R5                 ; Get first forward link.
     56   55   D1    323F 10166          CMPL    R5,R6                   ; Any (more) pool waiters?
          05   13    3242 10167          BEQL    POOL_WAITER_LOOP_END    ; Branch if not.
       18 A8   D6    3244 10168          INCL    MB$L_PL_WAITERS(R8)     ; Increment number of pool waiters on
                     3247 10169                                          ;  this port.
```

```
                F3   11  3247 10170          BRB     POOL_WAITER_LOOP        ; Loop till all pool waiters found.
                         3249 10171
                         3249 10172 POOL_WAITER_LOOP_END:
                         3249 10173
                         3249 10174 ;
                         3249 10175 ; Count the number of entries in the interlocked queue and store in MISC_BUF.
                         3249 10176 ;
                         3249 10177 ; INPUTS:
                         3249 10178 ;
                         3249 10179 ;       R8  -   Pointer into MISC_BUF to store data for the port.
                         3249 10180 ;       R7  -   Pointer to PDT.
                         3249 10181 ;
                         3249 10182 ; REGISTER USAGE:
                         3249 10183 ;
                         3249 10184 ;       R6  -   Pointer to queue to be counted.
                         3249 10185 ;       R7  -   Pointer to PDT.
                         3249 10186 ;       R8  -   Pointer to MISC_BUF.
                         3249 10187 ;       R9  -   Count of entries returned form CHECK_QUE routine.
                         3249 10188 ;
                         3249 10189
                         3249 10190 INTERLOCK_SNAP:
          56   01D0 C7   DE  3249 10191          MOVAL   PDT$Q_DFREEQ(R7),R6     ; Get address of datagram free queue.
                    59   D4  324E 10192          CLRL    R9                      ; Initialize count of entries in que.
                  00A1   30  3250 10193          BSBW    CHECK_QUE               ; Get the number of entries.
                  0082   30  3253 10194          BSBW    CNVRT_ASCII             ; Convert result to ascii string.
            1C A8    59   B0  3256 10195          MOVW    R9,MB$W_DG_FREE(R8)     ; Save number of free datagrams on this
                         325A 10196                                              ;  port.
          56   01D8 C7   DE  325A 10197          MOVAL   PDT$Q_MFREEQ(R7),R6     ; Get the address of the message free
                         325F 10198                                              ;  queue.
                    59   D4  325F 10199          CLRL    R9                      ; Initialize count of entries in que.
                  0090   30  3261 10200          BSBW    CHECK_QUE               ; Get the number of entries.
                    72   10  3264 10201          BSB     CNVRT_ASCII             ; Convert result to ascii string.
            1E A8    59   B0  3266 10202          MOVW    R9,MB$W_MSG_FREE(R8)    ; Save number of free messages on
                         326A 10203                                              ;  this port.
          56   01E0 C7   DE  326A 10204          MOVAL   PDT$Q_COMQL(R7),R6      ; Get address of low priority command
                         326F 10205                                              ;  queue.
                    59   D4  326F 10206          CLRL    R9                      ; Initialize count of entries in que.
                  0080   30  3271 10207          BSBW    CHECK_QUE               ; Get the number of entries in the low
                         3274 10208                                              ;  priority command queue.
          56   01E8 C7   DE  3274 10209          MOVAL   PDT$Q_COMQH(R7),R6      ; Get the address of the high priority
                         3279 10210                                              ;  command queue.
                    79   10  3279 10211          BSB     CHECK_QUE               ; Get the number of entries in the high
                         327B 10212                                              ;  priority command queue and add to
                         327B 10213                                              ;  total number of commands queued.
          56   01F0 C7   DE  327B 10214          MOVAL   PDT$Q_COMQ2(R7),R6      ; Get the address of command queue 2.
                    72   10  3280 10215          BSB     CHECK_QUE               ; Get the number of entries in command
                         3282 10216                                              ;  queue 2 and add to total number of
                         3282 10217                                              ;  commands qued.
          56   01F8 C7   DE  3282 10218          MOVAL   PDT$Q_COMQ3(R7),R6      ; Get the address of command queue 3.
                    6B   10  3287 10219          BSB     CHECK_QUE               ; Get the number of entries in command
                         3289 10220                                              ;  queue 3 and add to total number of
                         3289 10221                                              ;  commands qued.
                    4D   10  3289 10222          BSB     CNVRT_ASCII             ; Convert result to ascii string.
            20 A8    59   B0  328B 10223          MOVW    R9,MB$W_CMDS_QUED(R8)   ; Save the total number of commands
                         328F 10224                                              ;  queue on this port.
          56   0200 C7   DE  328F 10225          MOVAL   PDT$Q_RSPQ(R7),R6       ; Get the address of the response
                         3294 10226                                              ;  queue.
```

```
                     59   D4  3294 10227          CLRL    R9                          ; Initialize count of entries in que.
                     5C   10  3296 10228          BSB     CHECK_QUE                   ; Get the number of entries in the
                              3298 10229                                              ;  response queue.
                     3E   10  3298 10230          BSB     CNVRT_ASCII                 ; Convert result to ascii string.
                22 A8 59   B0  329A 10231          MOVW    R9,MB$W_RSPS_QUED(R8)       ; Save the number of responses queued
                              329E 10232                                              ;  on this port.
                              329E 10233
                              329E 10234 ;
                              329E 10235 ; Count the number of free buffer descriptors and save in NUM_FREE_BUF_DSCR.
                              329E 10236 ; Calculate the number of buffer descriptors in use and save in NUM_BUF_DSCR.
                              329E 10237 ;
                              329E 10238 ; REGISTER USAGE:
                              329E 10239 ;
                              329E 10240 ;       R6  -   Counter for number of free buffer descriptors.
                              329E 10241 ;       R7  -   Pointer to free buffer descriptors.
                              329E 10242 ;
                              329E 10243
                              329E 10244 FREE_BUF_SNAP:
          57 00000000'GF      D0   329E 10245          MOVL    G^SCS$GL_BDT,R7             ; Get the address of the buffer
                              32A5 10246                                              ;  descriptor table header.
          57 FFFFFFE8 8F      C0   32A5 10247          ADDL    #CIBDT$L_FREEBD-CIBD$L_LINK,R7
                              32AC 10248                                              ; Get the link to the first buffer
                              32AC 10249                                              ;  descriptor.
                     56   D4  32AC 10250          CLRL    R6                          ; Assume no free buffer descriptors.
                              32AE 10251
                              32AE 10252 FREE_BUF_LOOP:
             57   0C A7      D0   32AE 10253          MOVL    CIBD$L_LINK(R7),R7          ; Are there any (more) free buffer
                              32B2 10254                                              ;  descriptors?
                     04   13  32B2 10255          BEQL    FREE_BUF_END                ; Branch if not.
                     56   D6  32B4 10256          INCL    R6                          ; Increase count of free buffer
                              32B6 10257                                              ;  descriptors for this port.
                     F6   11  32B6 10258          BRB     FREE_BUF_LOOP               ; Loop for all free buffer descriptors.
                              32B8 10259
                              32B8 10260 FREE_BUF_END:
     0000000F'EF      56   D0  32B8 10261          MOVL    R6,NUM_FREE_BUF_DSCR        ; Save number of free buffer
                              32BF 10262                                              ;  descriptors.
          57 00000000'GF      3C   32BF 10263          MOVZWL  G^SCS$GW_BDTCNT,R7          ; Get the total number of buffer
                              32C6 10264                                              ;  descriptors available.
0000000B'EF   57      56    C3   32C6 10265          SUBL3   R6,R7,NUM_BUF_DSCR          ; Save the number of buffer descriptors
                              32CE 10266                                              ;  in use.
                        FE23  31   32CE 10267          BRW     DDB_LOOP                    ; Loop for all ports.
                              32D1 10268
                              32D1 10269 PORT_SNAP_END:
                              32D1 10270
                              32D1 10271 SNAPSHOT_END:
                     50   01  D0   32D1 10272          MOVL    #SS$_NORMAL,R0              ; Indicate success.
                              32D4 10273
                              32D4 10274 SNAPSHOT_ERR:
                              32D4 10275          ENBINT                              ; Lower IPL.
                     04       32D7 10276          RET                                 ; And return.
                              32D8 10277
```

```
                     32D8 10279 ;++
                     32D8 10280 ;
                     32D8 10281 ; FUNCTIONAL DESCRIPTION:
                     32D8 10282 ;
                     32D8 10283 ;       This routine converts the number of entries found in an interlock queue
                     32D8 10284 ;       to an ascii string.  If the number of entries found exceeds QUEUE_LIMIT
                     32D8 10285 ;       then a carrot ">" is placed before the count when it is output.
                     32D8 10286 ;
                     32D8 10287 ; CALLING SEQUENCE:
                     32D8 10288 ;
                     32D8 10289 ;       JSB     CNVRT_ASCII
                     32D8 10290 ;
                     32D8 10291 ; CALLED BY:
                     32D8 10292 ;
                     32D8 10293 ;       INTERLOCK_SNAP
                     32D8 10294 ;
                     32D8 10295 ; INPUTS:
                     32D8 10296 ;
                     32D8 10297 ;       Number of entries in the queue in R9.
                     32D8 10298 ;
                     32D8 10299 ; OUTPUTS:
                     32D8 10300 ;
                     32D8 10301 ;       Ascii string returned in R9.
                     32D8 10302 ;
                     32D8 10303 ; RETURN STATUS:
                     32D8 10304 ;
                     32D8 10305 ;       None.
                     32D8 10306 ;--
                     32D8 10307
                     32D8 10308 CNVRT_ASCII:
      59   03   91   32D8 10309          CMPB    #QUE_LIMIT,R9           ; Does the result equal the limit.
           0B   13   32DB 10310          BEQL    10$                     ; Branch if so.
59    59   08   78   32DD 10311          ASHL    #8,R9,R9                ; Get the count into the second byte.
59    3020 8F   A0   32E1 10312          ADDW    #^A/ 0/,R9              ; Convert second byte to ascii and make
                     32E6 10313                                          ;  the first byte a blank.
           0B   11   32E6 10314          BRB     20$                     ; Branch to common code.
           59   D7   32E8 10315 10$:     DECL    R9                      ; 3 becomes >2.
59    59   08   78   32EA 10316          ASHL    #8,R9,R9                ; Get the count into the second byte.
59    303E 8F   A0   32EE 10317          ADDW    #^A/>0/,R9              ; Convert second byte to ascii and make
                     32F3 10318                                          ;  the first byte a ">"
                05   32F3 10319 20$:     RSB
                     32F4 10320
```

```
                     32F4 10322          .SBTTL          -       Interlock Queue routine.
                     32F4 10323 ;++
                     32F4 10324 ;
                     32F4 10325 ; FUNCTIONAL DESCRIPTION:
                     32F4 10326 ;
                     32F4 10327 ;       This routine is used to count the number of entries in an interlocked
                     32F4 10328 ;        queue.  The number of entries found is returned in R9.  Note that R9
                     32F4 10329 ;        is not cleared by this routine so that R9 can be used to count the
                     32F4 10330 ;        total number of entries in several queues if this routine is called
                     32F4 10331 ;        repeatedly.  Since this routine executes at IPL 31 the maximum number
                     32F4 10332 ;        of entries counted by this routine has been limited to 3.
                     32F4 10333 ;
                     32F4 10334 ; CALLING SEQUENCE:
                     32F4 10335 ;
                     32F4 10336 ;       JSB CHECK_QUE
                     32F4 10337 ;
                     32F4 10338 ; CALLED BY:
                     32F4 10339 ;
                     32F4 10340 ;       INTERLOCK_SNAP
                     32F4 10341 ;
                     32F4 10342 ; INPUTS:
                     32F4 10343 ;
                     32F4 10344 ;       R6  = Address of queue for which entries are to be counted.
                     32F4 10345 ;
                     32F4 10346 ; OUTPUTS:
                     32F4 10347 ;
                     32F4 10348 ;       The number of entries in the queue is added to R9.
                     32F4 10349 ;
                     32F4 10350 ; IMPLICIT OUTPUTS:
                     32F4 10351 ;
                     32F4 10352 ;       R5 scratched.
                     32F4 10353 ;
                     32F4 10354 ; RETURN STATUS:
                     32F4 10355 ;
                     32F4 10356 ;       NORMAL or SS$_INTERLOCK status returned in R0.
                     32F4 10357 ;
                     32F4 10358 ; REGISTER USAGE:
                     32F4 10359 ;
                     32F4 10360 ;       R1 -    Interlock retry counter.
                     32F4 10361 ;       R4 -    Offset to next queue entry.
                     32F4 10362 ;       R5 -    Address of queue entry.
                     32F4 10363 ;       R6 -    Address of first entry in the queue.
                     32F4 10364 ;       R9 -    Counter for number of entries in the queue.
                     32F4 10365 ;
                     32F4 10366 ;--
                     32F4 10367
                     32F4 10368 CHECK_QUE:
  50   0000038C 8F    D0  32F4 10369          MOVL    #SS$_INTERLOCK,R0       ; Assume interlock failure.
                 51   D4  32FB 10370          CLRL    R1                      ; Initialize interlock retry counter.
            55   56    D0  32FD 10371          MOVL    R6,R5                   ; Save copy of queue address in R5.
                           3300 10372          DSBINT                          ; Disable interupts.
       54   65   01    CB  3306 10373          BICL3   #1,(R5),R4              ; Clear interlock bit in first
                           330A 10374                                          ;  self-relative offset.
                           330A 10375
                           330A 10376 WAS_LOCKED:
02 51  00000000'GF    F2  330A 10377          AOBLSS  G^EXE$GL_LOCKRTRY,R1,-  ; Try to lock again if we still have
                           3312 10378                  TRY_LOCK                ;  time.
```

```
                1B   11   3312 10379                BRB     LOCK_FAIL                       ; Otherwise, return with failure.
                          3314 10380
                          3314 10381 TRY_LOCK:
F2 66           00   E6   3314 10382                BBSSI   #0,(R6),WAS_LOCKED              ; Try to lock the queue.
                          3318 10383
 55   6544      9E   3318 10384 LOOP:    MOVAB   (R5)[R4],R5                             ; Get forward link to next entry.
   56   55      D1   331C 10385                    CMPL    R5,R6                           ; Does this link point back to the
                     331F 10386                                                          ;  first link?
                07   13   331F 10387                BEQL    10$                             ; Branch if it does.
   54   65      D0   3321 10388                    MOVL    (R5),R4                         ; Get next relative offset.
F0 59   03      F2   3324 10389                    AOBLSS  #QUE_LIMIT,R9,LOOP              ; Increment number of entries and
                     3328 10390                                                          ;  loop if limit not exceeded.
                     3328 10391
00 66   00      E7   3328 10392 10$:     BBCCI   #0,(R6),LOWERIPL                        ; Unlock the queue.
                     332C 10393
                     332C 10394 LOWERIPL:
   50   01      D0   332C 10395                    MOVL    #SS$_NORMAL,R0                  ; Indicate success.
                     332F 10396
                     332F 10397 LOCK_FAIL:
                     332F 10398                    ENBINT                                  ; Enable interupts.
                05   3332 10399                    RSB
                     3333 10400
                     3333 10401 ;NAP_END:
                     3333 10402
                     3333 10403
```

```
                     000004A3 10405          .PSECT  SNAP_DATA,RD,WRT,NOEXE
                         04A3 10406
                         04A3 10407 ;
                         04A3 10408 ; This buffer has been placed here so that the fixed length snapshot buffers
                         04A3 10409 ; can be locked in memory at the same time as the snapshot routines.  The
                         04A3 10410 ; program will not assemble if this buffer is placed before the snapshot
                         04A3 10411 ; buffers.  (For some reason FIX_BUF_LEN cannot be resolved)
                         04A3 10412 ;
                         04A3 10413
                         04A3 10414 OLD_FIX_BUF:                                 ; C- , of fix length buffer from
                000008FB 04A3 10415          .BLKB   FIX_BUF_LEN                ;  previous snapshot.  No need to
                         08FB 10416                                             ;  lock this buffer.
```

```
                        .SBTTL          -       GET_NUM_PB
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is used to count the number of circuits on a system.
                ;       It uses the information stored in PB buffer by the SNAPSHOT
                ;       routine to count the path blocks.  The number of circuits is stored in
                ;       the associated system block.
                ;
                ;       This code is separated from the snapshot routine to minimize time spent
                ;       at raised IPL.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       JSB GET_NUM_PB
                ;
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       SNAPSHOT routine has been executed and the snapshot data is stored in
                ;       SNAP_BUF.
                ;
                ; OUTPUTS:
                ;
                ;       Number of path blocks on each system block stored in the respective
                ;       system block at offset SB$SHC$L_NUM_PB.
                ;
                ; REGISTER USAGE:
                ;
                ;       R2  -   Counts the number of path blocks.
                ;       R10 -   Points to the first longword of a path block in PB buffer.
                ;       R11 -   Points to the first longword of a connection descriptor in
                ;               CDT buffer.
                ;
                ;--
                GET_NUM_PB:
      0C04 8F   BB              PUSHR   #^M<R2,R10,R11>
5B    000003ED'EF DO            MOVL    VAR_BUF,R11             ; Get address of first system block.
           52   D4              CLRL    R2                      ; Clear counter for number of paths on
                                                                ;  this system block.
      5A   0C AB   D0           MOVL    SB$L_PBFL(R11),R10      ; Get address of first path block on
                                                                ;  this system block.
           0D   13              BEQL    40$                     ; Branch if there are no circuits on
                                                                ;  this system.
      12   AA   B1    20$:     CMPW    PB$W_STATE(R10),-       ; Is the circuit in the open state?
           03                           #PB$C_OPEN              ;
           02   12              BNEQ    30$                     ; If not don't count it.
           52   D6              INCL    R2                      ; Increase count of path blocks on this
                                                                ;  system block.
      5A   6A   D0    30$:     MOVL    (R10),R10               ; Get address of next path block.
           F3   12              BNEQ    20$                     ; Branch if no more circuits on this
                                                                ;  system.
04 AB      52   D0    40$:     MOVL    R2,SB$SHC$L_NUM_PB(R11) ; Save number of paths on the system
                                                                ;  block.
      5B   6B   D0              MOVL    (R11),R11               ; Get address of next system block.
```

```
                E2  12  0922 10475          BNEQ    10$                     ; Loop for all systems in the cluster.
0C04            8F  BA  0924 10476          POPR    #^M<R2,R10,R11>
                    05  0928 10477          RSB
                        0929 10478
```

```
0929 10480          .SBTTL  Parser State Table.
0929 10481
0929 10482          $INIT_STATE     CMD_STATE,CMD_KEY
0929 10483
0929 10484          $STATE  CMD_START
0929 10485
0929 10486          $TRAN   '!',TPA$_EXIT               ; Ignore comments.
0929 10487
0929 10488          $TRAN   'SHOW',SHOW                 ; Command "SHOW".
0929 10489
0929 10490          $TRAN   'ADD',ADD                   ; Command "ADD".
0929 10491
0929 10492          $TRAN   'REMOVE',REMOVE             ; Command "REMOVE".
0929 10493
0929 10494          $TRAN   'SET',SET                   ; Command "SET".
0929 10495
0929 10496          $TRAN   'HELP',HELP                 ; Command "HELP".
0929 10497
0929 10498          $TRAN   'EXIT',TPA$_EXIT,-          ; Command "EXIT".
0929 10499                  CNTRLC_AST_DEL              ;
0929 10500
0929 10501          $TRAN   'INITIALIZE',TPA$_EXIT,-; Command "INITIALIZE".
0929 10502                  RESET
0929 10503
0929 10504          $TRAN   '@',TPA$_EXIT,EXECUTE       ; Execute procedure command.
0929 10505
0929 10506 ;
0929 10507 ; Translate the SHOW CLUSTER and SHOW LOCAL commands.
0929 10508 ;
0929 10509          $STATE  SHOW
0929 10510
0929 10511          $TRAN   TPA$_EOS,TPA$_EXIT,-        ; Return if end of string.
0929 10512                  REFRESH
0929 10513
0929 10514          $TRAN   'CLUSTER',TPA$_EXIT,-       ; "SHOW CLUSTER".
0929 10515                  SHOW_CLUSTER                ;
0929 10516
0929 10517          $TRAN   'LOCAL_PORTS',-             ; "SHOW LOCAL_PORTS".
0929 10518                  TPA$_EXIT,SHOW_LOCAL        ;
0929 10519
0929 10520
```

```
0929 10522 ;
0929 10523 ; Translate the add "class" or "field-name" commands.
0929 10524 ;
0929 10525          $STATE  ADD
0929 10526
0929 10527          $TRAN   TPA$_EOS,TPA$_EXIT,-    ; Return if end of string.
0929 10528                  REFRESH
0929 10529
0929 10530          $TRAN   'CLUSTER',ADD_CLUSTER,- ; "ADD CLUSTER"
0929 10531                  ,,TKN_DSCR              ;
0929 10532
0929 10533          $TRAN   'SYSTEMS',ADD_SYS,-     ; "ADD SYSTEMS".
0929 10534                  ,,TKN_DSCR              ;
0929 10535
0929 10536          $TRAN   'MEMBERS',ADD_MEMBERS,- ; "ADD MEMBERS.
0929 10537                  ,,TKN_DSCR              ;
0929 10538
0929 10539          $TRAN   'CONNECTIONS',ADD_CON,- ; "ADD CONNECTIONS".
0929 10540                  ,,TKN_DSCR              ;
0929 10541
0929 10542          $TRAN   'CIRCUITS',ADD_CIR,-    ; "ADD CIRCUITS".
0929 10543                  ,,TKN_DSCR              ;
0929 10544
0929 10545          $TRAN   'COUNTERS',-            ; "ADD COUNTERS".
0929 10546                  ADD_COUNTERS,-          ;
0929 10547                  ,,TKN_DSCR              ;
0929 10548
0929 10549          $TRAN   'CREDITS',ADD_CREDITS,- ; "ADD CREDITS".
0929 10550                  ,,TKN_DSCR              ;
0929 10551
0929 10552          $TRAN   'LOCAL_PORTS',-         ; "ADD LOCAL_PORTS".
0929 10553                  ADD_LOCAL_PORTS,-       ;
0929 10554                  ,,TKN_DSCR              ;
0929 10555
0929 10556          $TRAN   'ERRORS',ADD_ERRORS,-   ; "ADD ERRORS".
0929 10557                  ,,TKN_DSCR              ;
0929 10558
0929 10559          $TRAN   TPA$_SYMBOL,,ADD_FIELD  ; "ADD field-name".
0929 10560
0929 10561          $STATE  ADD_NEXT_CLASS
0929 10562          $TRAN   <',">,ADD               ; Loop if another class was
0929 10563                                          ;  specified.
0929 10564          $TRAN   '!',TPA$_EXIT,REFRESH   ; Allow comments...
0929 10565          $TRAN   TPA$_EOS,TPA$_EXIT,-    ;  or end of string.
0929 10566                  REFRESH                 ;
0929 10567          $TRAN   TPA$_LAMBDA,TPA$_FAIL,- ;
0929 10568                  REFRESH                 ; Anything else is an error.
0929 10569
```

```
0929 10571 ;
0929 10572 ; ADD CLUSTER
0929 10573 ;
0929 10574
0929 10575          $STATE  ADD_CLUSTER
0929 10576
0929 10577          $TRAN   '/',ADD_CLUSTER_ALL
0929 10578
0929 10579          $TRAN   TPA$_LAMBDA,-
0929 10580                  ADD_NEXT_CLASS,-        ;
0929 10581                  INCLUDE_CLASS,,,-       ;
0929 10582                  CLS$M_CLUB              ;
0929 10583
0929 10584          $STATE  ADD_CLUSTER_ALL
0929 10585          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD CLUSTER/ALL
0929 10586                  ADD_ALL_FIELDS,,,-      ;
0929 10587                  CLS$M_CLUB              ;
0929 10588
0929 10589 ;
0929 10590 ; ADD MEMBERS
0929 10591 ;
0929 10592
0929 10593          $STATE  ADD_MEMBERS
0929 10594
0929 10595          $TRAN   '/',ADD_MEMBERS_QUAL
0929 10596
0929 10597          $TRAN   TPA$_LAMBDA,-
0929 10598                  ADD_NEXT_CLASS,-        ;
0929 10599                  INCLUDE_CLASS,,,-       ;
0929 10600                  CLS$M_CSB               ;
0929 10601
0929 10602          $STATE  ADD_MEMBERS_QUAL
0929 10603          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD MEMBERS/ALL
0929 10604                  ADD_ALL_FIELDS,,,-      ;
0929 10605                  CLS$M_CSB               ;
0929 10606
0929 10607 ;
0929 10608 ; ADD COUNTERS
0929 10609 ;
0929 10610
0929 10611          $STATE  ADD_COUNTERS
0929 10612
0929 10613          $TRAN   '/',ADD_COUNTERS_QUAL
0929 10614
0929 10615          $TRAN   TPA$_LAMBDA,-
0929 10616                  ADD_NEXT_CLASS,-        ;
0929 10617                  INCLUDE_CLASS,,,-       ;
0929 10618                  CLS$M_CRT               ;
0929 10619
0929 10620          $STATE  ADD_COUNTERS_QUAL
0929 10621          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD COUNTERS/ALL
0929 10622                  ADD_ALL_FIELDS,,,-      ;
0929 10623                  CLS$M_CRT               ;
0929 10624
0929 10625
0929 10626 ;
0929 10627 ; ADD CREDITS
```

```
0929 10628 ;
0929 10629
0929 10630          $STATE  ADD_CREDITS
0929 10631
0929 10632          $TRAN   '/',ADD_CREDITS_QUAL
0929 10633
0929 10634          $TRAN   TPA$_LAMBDA,-
0929 10635                  ADD_NEXT_CLASS,-        ;
0929 10636                  INCLUDE_CLASS,,,-       ;
0929 10637                  CLS$M_CRD               ;
0929 10638
0929 10639          $STATE  ADD_CREDITS_QUAL
0929 10640          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD CREDITS/ALL
0929 10641                  ADD_ALL_FIELDS,,,-      ;
0929 10642                  CLS$M_CRD               ;
0929 10643
0929 10644
0929 10645 ;
0929 10646 ; ADD ERRORS
0929 10647 ;
0929 10648
0929 10649          $STATE  ADD_ERRORS
0929 10650
0929 10651          $TRAN   '/',ADD_ERRORS_QUAL
0929 10652
0929 10653          $TRAN   TPA$_LAMBDA,-
0929 10654                  ADD_NEXT_CLASS,-        ;
0929 10655                  INCLUDE_CLASS,,,-       ;
0929 10656                  CLS$M_ERR               ;
0929 10657
0929 10658          $STATE  ADD_ERRORS_QUAL
0929 10659          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD ERRORS/ALL
0929 10660                  ADD_ALL_FIELDS,,,-      ;
0929 10661                  CLS$M_ERR               ;
0929 10662
0929 10663 ;
0929 10664 ; ADD LOCAL_PORTS
0929 10665 ;
0929 10666
0929 10667          $STATE  ADD_LOCAL_PORTS
0929 10668
0929 10669          $TRAN   '/',ADD_LOCAL_PORTS_QUAL
0929 10670
0929 10671          $TRAN   TPA$_LAMBDA,-
0929 10672                  ADD_NEXT_CLASS,-        ;
0929 10673                  INCLUDE_CLASS,,,-       ;
0929 10674                  CLS$M_LPRT              ;
0929 10675
0929 10676          $STATE  ADD_LOCAL_PORTS_QUAL
0929 10677          $TRAN   'ALL',ADD_NEXT_CLASS,-  ; ADD LOCAL_PORTS/ALL
0929 10678                  ADD_ALL_FIELDS,,,-      ;
0929 10679                  CLS$M_LPRT              ;
0929 10680
0929 10681
```

```
0929 10683 ;
0929 10684 ; ADD SYSTEMS
0929 10685 ;
0929 10686          $STATE  ADD_SYS
0929 10687
0929 10688          $TRAN   '/',ADD_SYS_QUAL        ; Go to ADD_SYS_QUAL state if qualifier
0929 10689                                          ;  is present.
0929 10690
0929 10691          $TRAN   TPA$_LAMBDA,-           ; Otherwise just ADD SYSTEMS.
0929 10692                  ADD_NEXT_CLASS,-        ;
0929 10693                  INCLUDE_CLASS,,,-       ;
0929 10694                  CLS$M_SYS               ;
0929 10695
0929 10696 ;
0929 10697 ; Process ADD SYSTEMS qualifiers.
0929 10698 ;
0929 10699          $STATE  ADD_SYS_QUAL
0929 10700
0929 10701          $TRAN   'ALL',-                 ; ADD SYSTEMS/ALL
0929 10702                  NEXT_ADD_SYS_QUAL,-     ;
0929 10703                  ADD_ALL_FIELDS,,,-      ;
0929 10704                  CLS$M_SYS               ;
0929 10705          $TRAN   'ID',ADD_SYS_ID         ; ADD SYSTEM/ID
0929 10706          $TRAN   'TYPE',ADD_SYS_TYPE     ; ADD SYSTEM/TYPE
0929 10707          $TRAN   'NODE',ADD_SYS_NODE     ; ADD SYSTEM/NODE
0929 10708          $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 10709
0929 10710 ;
0929 10711 ; Loop for multiple qualifiers on the SYSTEMS class.
0929 10712 ;
0929 10713          $STATE NEXT_ADD_SYS_QUAL
0929 10714          $TRAN   '/',ADD_SYS_QUAL
0929 10715          $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929 10716
0929 10717
```

```
0929 10719 ;
0929 10720 ; Process ADD SYSTEMS/ID=system_id
0929 10721 ;         ADD SYSTEMS/ID=(system_id,...)
0929 10722
0929 10723          $STATE  ADD_SYS_ID
0929 10724          $TRAN   '='
0929 10725          $STATE
0929 10726          $TRAN   '(',ADD_ID_LOOP          ; Go parse a list of ID's
0929 10727          $TRAN   TPA$_LAMBDA,             ; Go parse a single ID
0929 10728          $STATE
0929 10729          $TRAN   !ADD_SYS_ID_STRING,-     ;
0929 10730                  NEXT_ADD_SYS_QUAL        ;
0929 10731
0929 10732          $STATE  ADD_ID_LOOP              ; Parse a list of ID numbers till a ")"
0929 10733                                           ;  is found.
0929 10734          $TRAN   !ADD_SYS_ID_STRING       ;
0929 10735          $STATE
0929 10736          $TRAN   <',',>,ADD_ID_LOOP       ; Go parse a single ID
0929 10737          $TRAN   ')',NEXT_ADD_SYS_QUAL    ; Exit loop if ")" found.
0929 10738
0929 10739          $STATE  ADD_SYS_ID_STRING        ; Parse a single ID
0929 10740
0929 10741          $TRAN   TPA$_STRING,,,,TKN_DSCR ; Save address of string.
0929 10742          $STATE
0929 10743          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Call REM_ID to remove system from
0929 10744                  REM_ID                   ;  disabled list.
0929 10745
0929 10746
0929 10747
```

```
0929 10749 ;
0929 10750 ; Process ADD SYSTEMS/TYPE=hw_type
0929 10751 ;         ADD SYSTEMS/TYPE=(hw_type,...)
0929 10752
0929 10753          $STATE  ADD_SYS_TYPE
0929 10754          $TRAN   '='
0929 10755          $STATE
0929 10756          $TRAN   '(',ADD_SYS_TYPE_LOOP   ; Go parse a list of system types.
0929 10757          $TRAN   TPA$_LAMBDA,            ; Go parse a single system type.
0929 10758          $STATE
0929 10759          $TRAN   !ADD_SYS_TYPE_STRING,-;
0929 10760                  NEXT_ADD_SYS_QUAL       ;
0929 10761
0929 10762          $STATE  ADD_SYS_TYPE_LOOP       ; Parse a list of system types till a
0929 10763                                          ;  ")" is found.
0929 10764          $TRAN   !ADD_SYS_TYPE_STRING    ;
0929 10765          $STATE
0929 10766          $TRAN   <',',>,ADD_SYS_TYPE_LOOP ; Go parse a single system type.
0929 10767          $TRAN   ')',NEXT_ADD_SYS_QUAL   ; Exit loop if ")" found.
0929 10768
0929 10769
0929 10770          $STATE  ADD_SYS_TYPE_STRING     ; Parse a single system type
0929 10771
0929 10772          $TRAN   'ALL',TPA$_EXIT,-       ; "ADD SYSTEMS/TYPE=ALL" - Add all
0929 10773                  INCLUDE_CLASS,,,-       ; types of systems to the display.
0929 10774                  CLS$M_ALL_SYS
0929 10775
0929 10776          $TRAN   'V780',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=780" - Add only
0929 10777                  INCLUDE_CLASS,,,-       ;   11/780 systems to the display.
0929 10778                  CLS$M_780_SYS           ;
0929 10779
0929 10780          $TRAN   'V785',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=785" - Add only
0929 10781                  INCLUDE_CLASS,,,-       ;   11/785 systems to the display.
0929 10782                  CLS$M_785_SYS           ;
0929 10783
0929 10784          $TRAN   'V750',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=750" - Add only
0929 10785                  INCLUDE_CLASS,,,-       ;   11/750 systems to the display.
0929 10786                  CLS$M_750_SYS           ;
0929 10787
0929 10788          $TRAN   'HS50',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=HS50" - Add only
0929 10789                  INCLUDE_CLASS,,,-       ;   HSC50 systems to the display.
0929 10790                  CLS$M_HSC50_SYS         ;
0929 10791
0929 10792          $TRAN   'HS60',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=HS60" - Add only
0929 10793                  INCLUDE_CLASS,,,-       ;   HSC60 systems to the display.
0929 10794                  CLS$M_HSC60_SYS         ;
0929 10795
0929 10796          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929 10797
0929 10798
```

```
                                    M 4
0929 10800 ;
0929 10801 ; Process ADD SYSTEMS/NODE=nodename
0929 10802 ;         ADD SYSTEMS/NODE=(nodename,...)
0929 10803
0929 10804          $STATE  ADD_SYS_NODE
0929 10805          $TRAN   '='
0929 10806          $STATE
0929 10807          $TRAN   '(',ADD_NODE_LOOP       ; Go parse a list of nodenames
0929 10808          $TRAN   TPA$_LAMBDA             ; Go parse a single nodename
0929 10809          $STATE
0929 10810          $TRAN   !ADD_SYS_NODE_STRING,-  ;
0929 10811                  NEXT_ADD_SYS_QUAL       ;
0929 10812
0929 10813          $STATE  ADD_NODE_LOOP           ; Parse a list of nodenames till a ")"
0929 10814                                          ;  is found.
0929 10815          $TRAN   !ADD_SYS_NODE_STRING    ; Go parse a nodename.
0929 10816          $STATE
0929 10817          $TRAN   <',',ADD_NODE_LOOP      ; Loop if "," found.
0929 10818          $TRAN   ')',NEXT_ADD_SYS_QUAL   ; Until a ")" is found.
0929 10819
0929 10820          $STATE  ADD_SYS_NODE_STRING     ; Parse a single nodename
0929 10821          $TRAN   TPA$_STRING,,,,TKN_DSCR ; Save address of string.
0929 10822          $STATE
0929 10823          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Call ADD_NODE.
0929 10824                  ADD_NODE
0929 10825
0929 10826
```

```
0929 10828 ;
0929 10829 ; ADD CIRCUITS
0929 10830 ;
0929 10831         $STATE  ADD_CIR
0929 10832
0929 10833         $TRAN   '/',ADD_CIR_QUAL        ; Go to ADD_CIR_QUAL state if qualifier
0929 10834                                         ;  is present.
0929 10835
0929 10836         $TRAN   TPA$_LAMBDA,-           ; Otherwise add all types of circuits
0929 10837                 ADD_NEXT_CLASS,-        ;  to the display.
0929 10838                 INCLUDE_CLASS,,,-       ;
0929 10839                 CLS$M_ALL_CIR           ;
0929 10840
0929 10841 ;
0929 10842 ; Process ADD CIRCUITS qualifiers.
0929 10843 ;
0929 10844         $STATE  ADD_CIR_QUAL
0929 10845
0929 10846         $TRAN   'ALL',-                 ; ADD CIRCUITS/ALL
0929 10847                 NEXT_ADD_CIR_QUAL,-     ;
0929 10848                 ADD_ALL_FIELDS,,,-      ;
0929 10849                 CLS$M_ALL_CIR
0929 10850         $TRAN   'TYPE',ADD_CIR_TYPE     ; ADD CIRCUITS/TYPE
0929 10851         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 10852
0929 10853 ;
0929 10854 ; Loop for multiple qualifiers on the CIRCUITS class.
0929 10855 ;
0929 10856         $STATE NEXT_ADD_CIR_QUAL
0929 10857         $TRAN   '/',ADD_CIR_QUAL
0929 10858         $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929 10859
0929 10860         $STATE  ADD_CIR_TYPE
0929 10861         $TRAN   '='
0929 10862         $STATE
0929 10863         $TRAN   '(',ADD_CIR_TYPE_LOOP   ; Go parse a list of circuit types.
0929 10864         $TRAN   TPA$_LAMBDA,            ; Go parse a single circuit type.
0929 10865         $STATE
0929 10866         $TRAN   !ADD_CIR_TYPE_STRING,-  ;
0929 10867                 NEXT_ADD_CIR_QUAL       ;
0929 10868
0929 10869         $STATE  ADD_CIR_TYPE_LOOP       ; Parse a list of circuit types till a
0929 10870                                         ;  ")" is found.
0929 10871         $TRAN   !ADD_CIR_TYPE_STRING    ;
0929 10872         $STATE
0929 10873         $TRAN   <',',>,ADD_CIR_TYPE_LOOP ; Go parse a single circuit type.
0929 10874         $TRAN   ')',NEXT_ADD_CIR_QUAL   ; Exit loop if ")" found.
0929 10875
0929 10876
0929 10877         $STATE  ADD_CIR_TYPE_STRING     ; Parse a single circuit type.
0929 10878
0929 10879         $TRAN   'ALL',TPA$_EXIT,-       ; "ADD CIRCUITS/TYPE=ALL" - Add all
0929 10880                 INCLUDE_CLASS,,,-       ; types of circuits to the display.
0929 10881                 CLS$M_ALL_CIR
0929 10882
0929 10883         $TRAN   'OPEN',TPA$_EXIT,-      ; "ADD CIRCUITS/OPEN" - Add circuits in
0929 10884                 INCLUDE_CLASS,,,-       ;  the open state to the display.
```

```
0929 10885                      CLS$M_CIR!CLS$M_OPEN_CIR;
0929 10886
0929 10887          $TRAN       'NOOPEN',TPA$_EXIT,-    ; ''ADD CIRCUITS/NOOPEN'' - Add circuits
0929 10888                      INCLUDE_CLASS,,,-       ;  in non-open states to the display.
0929 10889                      CLS$M_CIR!CLS$M_NOOPEN_CIR
0929 10890
0929 10891          $TRAN       TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929 10892
0929 10893
```

```
0929 10895 ;
0929 10896 ; ADD CONNECTIONS
0929 10897 ;
0929 10898         $STATE  ADD_CON
0929 10899
0929 10900         $TRAN   '/',ADD_CON_QUAL        ; Go to ADD_CON_QUAL state if qualifier
0929 10901                                         ;  is present.
0929 10902
0929 10903         $TRAN   TPA$_LAMBDA,-           ; Otherwise add all types of connections
0929 10904                 ADD_NEXT_CLASS,-        ;  to the display.
0929 10905                 INCLUDE_CLASS,,,-       ;
0929 10906                 CLS$M_ACL_CON           ;
0929 10907
0929 10908 ;
0929 10909 ; Process ADD CONNECTIONS qualifiers.
0929 10910 ;
0929 10911         $STATE  ADD_CON_QUAL
0929 10912
0929 10913         $TRAN   'ALL',-                 ; ADD CONNECTIONS/ALL
0929 10914                 NEXT_ADD_CON_QUAL,-     ;
0929 10915                 ADD_ALL_FIELDS,,,-      ;
0929 10916                 CLS$M_ACL_CON
0929 10917         $TRAN   'TYPE',ADD_CON_TYPE     ; ADD CONNECTIONS/TYPE
0929 10918         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 10919
0929 10920 ;
0929 10921 ; Loop for multiple qualifiers on the CONNECTIONS class.
0929 10922 ;
0929 10923         $STATE NEXT_ADD_CON_QUAL
0929 10924         $TRAN   '/',ADD_CON_QUAL
0929 10925         $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929 10926
0929 10927         $STATE  ADD_CON_TYPE
0929 10928         $TRAN   '='
0929 10929         $STATE
0929 10930         $TRAN   '(',ADD_CON_TYPE_LOOP   ; Go parse a list of connection types.
0929 10931         $TRAN   TPA$_LAMBDA,            ; Go parse a single connection type.
0929 10932         $STATE
0929 10933         $TRAN   !ADD_CON_TYPE_STRING,-  ;
0929 10934                 NEXT_ADD_CON_QUAL       ;
0929 10935
0929 10936
0929 10937         $STATE  ADD_CON_TYPE_LOOP       ; Parse a list of connection types till
0929 10938                                         ;  a ")" is found.
0929 10939         $TRAN   !ADD_CON_TYPE_STRING    ;
0929 10940         $STATE
0929 10941         $TRAN   <',',>,ADD_CON_TYPE_LOOP ; Go parse a single connection type.
0929 10942         $TRAN   ')',NEXT_ADD_CON_QUAL   ; Exit loop if ")" found.
0929 10943
0929 10944
0929 10945         $STATE  ADD_CON_TYPE_STRING     ; Parse a single connection type.
0929 10946
0929 10947         $TRAN   'ALL',TPA$_EXIT,-       ; "ADD CONNECTION/TYPE=ALL" - Add all
0929 10948                 INCLUDE_CLASS,,,-       ; types of connections to the display.
0929 10949                 CLS$M_ACL_CON
0929 10950
0929 10951         $TRAN   'OPEN',TPA$_EXIT,-      ; "ADD CONNECTIONS/OPEN" - Add
```

```
0929 10952                      INCLUDE_CLASS,,,-       ;  connections in the open state to the
0929 10953                      CLS$M_CON!CLS$M_OPEN_CON;  display.
0929 10954
0929 10955          $TRAN       'NOOPEN',TPA$_EXIT,-    ; ''ADD CONNECTIONS/NOOPEN'' - Add
0929 10956                      INCLUDE_CLASS,,,-       ;  connections in non-open states to
0929 10957                      CLS$M_CON!CLS$M_NOOPEN_CON  ;  the display.
0929 10958
0929 10959          $TRAN       TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929 10960
0929 10961
```

```
0929 10963 ;
0929 10964 ; Translate the REMOVE "class" or "field-name" commands.
0929 10965 ;
0929 10966          $STATE  REMOVE
0929 10967
0929 10968          $TRAN   TPA$_EOS,TPA$_EXIT,-    ; Return if end of string.
0929 10969                  REFRESH
0929 10970
0929 10971          $TRAN   'CLUSTER',REM_CLUSTER,- ; "REMOVE CLUSTER"
0929 10972                  ,,TKN_DSCR              ;
0929 10973
0929 10974          $TRAN   'SYSTEMS',REM_SYS,-     ; "REMOVE SYSTEMS".
0929 10975                  ,,TKN_DSCR              ;
0929 10976
0929 10977          $TRAN   'MEMBERS',REM_MEMBERS,- ; "REMOVE MEMBERS".
0929 10978                  ,,TKN_DSCR              ;
0929 10979
0929 10980          $TRAN   'CONNECTIONS',REM_CON,- ; "REMOVE CONNECTIONS".
0929 10981                  ,,TKN_DSCR              ;
0929 10982
0929 10983          $TRAN   'CIRCUITS',REM_CIR,-    ; "REMOVE CIRCUITS".
0929 10984                  ,,TKN_DSCR              ;
0929 10985
0929 10986          $TRAN   'COUNTERS',-            ; "REMOVE COUNTERS".
0929 10987                  REM_COUNTERS,-          ;
0929 10988                  ,,TKN_DSCR              ;
0929 10989
0929 10990          $TRAN   'CREDITS',REM_CREDITS,- ; "REMOVE CREDITS".
0929 10991                  ,,TKN_DSCR              ;
0929 10992
0929 10993          $TRAN   'ERRORS',REM_ERRORS,-   ; "REMOVE ERRORS".
0929 10994                  ,,TKN_DSCR              ;
0929 10995
0929 10996          $TRAN   'LOCAL_PORTS',-         ; "REMOVE LOCAL_PORTS".
0929 10997                  REM_LOCAL_PORTS,-       ;
0929 10998                  ,,TKN_DSCR              ;
0929 10999
0929 11000          $TRAN   TPA$_SYMBOL,,-          ; "REMOVE field-name".
0929 11001                  REMOVE_FIELD,-          ;
0929 11002                  ,,TKN_DSCR              ;
0929 11003
0929 11004
0929 11005          $STATE  REM_NEXT_CLASS
0929 11006          $TRAN   <',">,REMOVE            ; Loop if another class was
0929 11007                                          ;  specified.
0929 11008          $TRAN   '!',TPA$_EXIT,REFRESH   ; Allow comments...
0929 11009          $TRAN   TPA$_EOS,TPA$_EXIT,-    ;
0929 11010                  REFRESH                 ;  or end of string.
0929 11011          $TRAN   TPA$_LAMBDA,TPA$_FAIL,- ;
0929 11012                  REFRESH                 ; Anything else is an error.
0929 11013
0929 11014
```

```
0929 11016 ;
0929 11017 ; REMOVE CLUSTER
0929 11018 ;
0929 11019
0929 11020          $STATE  REM_CLUSTER                 ;
0929 11021          $TRAN   TPA$_LAMBDA,-               ;
0929 11022                  REM_NEXT_CLASS,-            ;
0929 11023                  EXCLUDE_CLASS,,,-           ;
0929 11024                  CLS$M_CLUB                  ;
0929 11025
0929 11026 ;
0929 11027 ; REMOVE MEMBERS
0929 11028 ;
0929 11029
0929 11030          $STATE  REM_MEMBERS                 ;
0929 11031          $TRAN   TPA$_LAMBDA,-               ;
0929 11032                  REM_NEXT_CLASS,-            ;
0929 11033                  EXCLUDE_CLASS,,,-           ;
0929 11034                  CLS$M_CSB                   ;
0929 11035
0929 11036 ;
0929 11037 ; REMOVE COUNTERS
0929 11038 ;
0929 11039          $STATE  REM_COUNTERS
0929 11040          $TRAN   TPA$_LAMBDA,-               ;
0929 11041                  REM_NEXT_CLASS,-            ;
0929 11042                  EXCLUDE_CLASS,,,-           ;
0929 11043                  CLS$M_CNT                   ;
0929 11044
0929 11045 ;
0929 11046 ; REMOVE CREDITS
0929 11047 ;
0929 11048          $STATE  REM_CREDITS
0929 11049          $TRAN   TPA$_LAMBDA,-               ;
0929 11050                  REM_NEXT_CLASS,-            ;
0929 11051                  EXCLUDE_CLASS,,,-           ;
0929 11052                  CLS$M_CRD                   ;
0929 11053
0929 11054 ;
0929 11055 ; REMOVE LOCAL_PORTS
0929 11056 ;
0929 11057
0929 11058          $STATE  REM_LOCAL_PORTS
0929 11059          $TRAN   TPA$_LAMBDA,-               ;
0929 11060                  REM_NEXT_CLASS,-            ;
0929 11061                  EXCLUDE_CLASS,,,-           ;
0929 11062                  CLS$M_LPRT                  ;
0929 11063
0929 11064 ;
0929 11065 ; REMOVE ERRORS
0929 11066 ;
0929 11067
0929 11068          $STATE  REM_ERRORS
0929 11069          $TRAN   TPA$_LAMBDA,-               ;
0929 11070                  REM_NEXT_CLASS,-            ;
0929 11071                  EXCLUDE_CLASS,,,-           ;
0929 11072                  CLS$M_ERR                   ;
```

```
0929 11074 ;
0929 11075 ; REMOVE SYSTEMS
0929 11076 ;
0929 11077          $STATE  REM_SYS
0929 11078
0929 11079          $TRAN   '/',REM_SYS_QUAL        ; Go to REM_SYS_QUAL state if
0929 11080                                          ;  qualifier is present.
0929 11081          $TRAN   TPA$_LAMBDA,-           ; Otherwise just REMOVE SYSTEMS.
0929 11082                  REM_NEXT_CLASS,-        ;
0929 11083                  EXCLUDE_CLASS,,,-       ;
0929 11084                  CLS$M_SYS               ;
0929 11085
0929 11086 ;
0929 11087 ; Process REMOVE SYSTEMS qualifiers.
0929 11088 ;
0929 11089          $STATE  REM_SYS_QUAL
0929 11090
0929 11091          $TRAN   'ID',REM_SYS_ID         ; REMOVE SYSTEM/ID
0929 11092          $TRAN   'TYPE',REM_SYS_TYPE     ; REMOVE SYSTEM/TYPE
0929 11093          $TRAN   'NODE',REM_SYS_NODE     ; REMOVE SYSTEM/NODE
0929 11094          $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 11095
0929 11096 ;
0929 11097 ; Loop for multiple qualifiers on the SYSTEMS class.
0929 11098 ;
0929 11099          $STATE NEXT_REM_SYS_QUAL
0929 11100          $TRAN   '/',REM_SYS_QUAL
0929 11101          $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929 11102
0929 11103
```

```
0929 11105 ;
0929 11106 ; Process REMOVE SYSTEMS/ID=system_id
0929 11107 ;         REMOVE SYSTEMS/ID=(system_id,...)
0929 11108 ;
0929 11109          $STATE  REM_SYS_ID
0929 11110          $TRAN   '='
0929 11111          $STATE
0929 11112          $TRAN   '(',REM_ID_LOOP          ; Go parse a list of ID's
0929 11113          $TRAN   TPA$_LAMBDA,             ; Go parse a single ID
0929 11114          $STATE
0929 11115          $TRAN   !REM_SYS_ID_STRING,-     ;
0929 11116                  NEXT_REM_SYS_QUAL        ;
0929 11117 ;
0929 11118          $STATE  REM_ID_LOOP              ; Parse a list of ID numbers till a ")"
0929 11119 ;                                         ;  is found.
0929 11120          $TRAN   !REM_SYS_ID_STRING       ;
0929 11121          $STATE
0929 11122          $TRAN   <',',>,REM_ID_LOOP       ; Go parse a single ID
0929 11123          $TRAN   ')',NEXT_REM_SYS_QUAL    ; Exit loop if ")" found.
0929 11124 ;
0929 11125          $STATE  REM_SYS_ID_STRING        ; Parse a single ID
0929 11126 ;
0929 11127          $TRAN   TPA$_STRING,,,,TKN_DSCR  ; Save address of string.
0929 11128          $STATE
0929 11129          $TRAN   TPA$_LAMBDA,TPA$_EXIT,-  ; Call ADD_ID to add system to
0929 11130                  ADD_ID                   ;  disabled list.
0929 11131 ;
0929 11132
```

```
0929 11134 ;
0929 11135 ; Process REMOVE SYSTEMS/TYPE=hw_type
0929 11136 ;         REMOVE SYSTEMS/TYPE=(hw_type,...)
0929 11137
0929 11138          $STATE  REM_SYS_TYPE
0929 11139          $TRAN   '='
0929 11140          $STATE
0929 11141          $TRAN   '(',REM_SYS_TYPE_LOOP   ; Go parse a list of system types.
0929 11142          $TRAN   TPA$_LAMBDA,            ; Go parse a single system type.
0929 11143          $STATE
0929 11144          $TRAN   !REM_SYS_TYPE_STRING,-  ;
0929 11145                  NEXT_REM_SYS_QUAL       ;
0929 11146
0929 11147          $STATE  REM_SYS_TYPE_LOOP       ; Parse a list of system types till a
0929 11148                                          ;  ")" is found.
0929 11149          $TRAN   !REM_SYS_TYPE_STRING    ;
0929 11150          $STATE
0929 11151          $TRAN   <',',>,REM_SYS_TYPE_LOOP ; Go parse a single system type.
0929 11152          $TRAN   ')',NEXT_REM_SYS_QUAL   ; Exit loop if ")" found.
0929 11153
0929 11154
0929 11155          $STATE  REM_SYS_TYPE_STRING     ; Parse a single system type
0929 11156
0929 11157          $TRAN   'ALL',TPA$_EXIT,-       ; "REMOVE SYSTEMS/TYPE=ALL" - Remove all
0929 11158                  EXCLUDE_CLASS,,,-       ; types of systems from the display.
0929 11159                  CLS$M_ALL_SYS
0929 11160
0929 11161          $TRAN   'V780',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=780" - Remove
0929 11162                  EXCLUDE_CLASS,,,-       ;  only 11/780 systems from the display.
0929 11163                  CLS$M_780_SYS           ;
0929 11164
0929 11165          $TRAN   'V785',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=785" - Remove
0929 11166                  EXCLUDE_CLASS,,,-       ;  only 11/785 systems from the display.
0929 11167                  CLS$M_785_SYS           ;
0929 11168
0929 11169          $TRAN   'V750',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=750" - Remove
0929 11170                  EXCLUDE_CLASS,,,-       ;  only 11/750 systems from the display.
0929 11171                  CLS$M_750_SYS           ;
0929 11172
0929 11173          $TRAN   'HS50',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=HS50" - Remove
0929 11174                  EXCLUDE_CLASS,,,-       ;  only HSC50 systems from the display.
0929 11175                  CLS$M_HSC50_SYS         ;
0929 11176
0929 11177          $TRAN   'HS60',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=HS60" - Remove
0929 11178                  EXCLUDE_CLASS,,,-       ;  only HSC60 systems from the display.
0929 11179                  CLS$M_HSC60_SYS         ;
0929 11180
0929 11181          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929 11182
0929 11183
```

```
                                                J 5
0929 11185 ;
0929 11186 ; Process REMOVE SYSTEMS/NODE=nodename
0929 11187 ;         REMOVE SYSTEMS/NODE=(nodename,...)
0929 11188
0929 11189          $STATE  REM_SYS_NODE
0929 11190          $TRAN   '='
0929 11191          $STATE
0929 11192          $TRAN   '(',REM_NODE_LOOP       ; Go parse a list of nodenames
0929 11193          $TRAN   TPA$_LAMBDA             ; Go parse a single nodename
0929 11194          $STATE
0929 11195          $TRAN   !REM_SYS_NODE_STRING,-  ;
0929 11196                  NEXT_REM_SYS_QUAL       ;
0929 11197
0929 11198          $STATE  REM_NODE_LOOP           ; Parse a list of nodenames till a ")"
0929 11199                                          ;  is found.
0929 11200          $TRAN   !REM_SYS_NODE_STRING    ; Go parse a nodename.
0929 11201          $STATE
0929 11202          $TRAN   <',',>,REM_NODE_LOOP    ; Loop if "," found.
0929 11203          $TRAN   ')',NEXT_REM_SYS_QUAL   ; Until a ")" is found.
0929 11204
0929 11205          $STATE  REM_SYS_NODE_STRING     ; Parse a single nodename
0929 11206          $TRAN   TPA$_STRING,,,,TKN_DSCR ; Save address of string.
0929 11207          $STATE
0929 11208          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Call REM_NODE.
0929 11209                  REM_NODE
0929 11210
```

```
0929 11212 ;
0929 11213 ; REMOVE CIRCUITS
0929 11214 ;
0929 11215          $STATE  REM_CIR
0929 11216
0929 11217          $TRAN   '/',REM_CIR_QUAL        ; Go to REM_CIR_QUAL state if qualifier
0929 11218                                          ;  is present.
0929 11219
0929 11220          $TRAN   TPA$_LAMBDA,-           ; Otherwise remove all types of circuits
0929 11221                  REM_NEXT_CLASS,-        ;  to the display.
0929 11222                  EXCLUDE_CLASS,,,-       ;
0929 11223                  CLS$M_ALL_CIR           ;
0929 11224
0929 11225 ;
0929 11226 ; Process REMOVE CIRCUITS qualifiers.
0929 11227 ;
0929 11228          $STATE  REM_CIR_QUAL
0929 11229
0929 11230          $TRAN   'TYPE',REM_CIR_TYPE     ; REMOVE CIRCUITS/TYPE
0929 11231          $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 11232
0929 11233 ;
0929 11234 ; Loop for multiple qualifiers on the CIRCUITS class.
0929 11235 ;
0929 11236          $STATE NEXT_REM_CIR_QUAL
0929 11237          $TRAN   '/',REM_CIR_QUAL
0929 11238          $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929 11239
0929 11240          $STATE  REM_CIR_TYPE
0929 11241          $TRAN   '='
0929 11242          $STATE
0929 11243          $TRAN   '(',REM_CIR_TYPE_LOOP   ; Go parse a list of circuit types.
0929 11244          $TRAN   TPA$_LAMBDA,            ; Go parse a single circuit type.
0929 11245          $STATE
0929 11246          $TRAN   !REM_CIR_TYPE_STRING,-  ;
0929 11247                  NEXT_REM_CIR_QUAL       ;
0929 11248
0929 11249
0929 11250          $STATE  REM_CIR_TYPE_LOOP       ; Parse a list of circuit types till a
0929 11251                                          ;  ")" is found.
0929 11252          $TRAN   !REM_CIR_TYPE_STRING    ;
0929 11253          $STATE
0929 11254          $TRAN   <',',>,REM_CIR_TYPE_LOOP ; Go parse a single circuit type.
0929 11255          $TRAN   ')',NEXT_REM_CIR_QUAL   ; Exit loop if ")" found.
0929 11256
0929 11257
0929 11258          $STATE  REM_CIR_TYPE_STRING     ; Parse a single circuit type.
0929 11259
0929 11260          $TRAN   'ALL',TPA$_EXIT,-       ; "REMOVE CIRCUITS/TYPE=ALL" - Remove
0929 11261                  EXCLUDE_CLASS,,,-       ;  all types of circuits from the
0929 11262                  CLS$M_ALL_CIR           ;  display.
0929 11263
0929 11264          $TRAN   'OPEN',TPA$_EXIT,-      ; "REMOVE CIRCUITS/OPEN" - Remove
0929 11265                  EXCLUDE_CLASS,,,-       ;  circuits in the open state from to
0929 11266                  CLS$M_OPEN_CIR          ;  the display.
0929 11267
0929 11268          $TRAN   'NOOPEN',TPA$_EXIT,-    ; "REMOVE CIRCUITS/NOOPEN" - Remove
```

```
0929 11269                      EXCLUDE_CLASS,,-         ;  circuits in non-open states from the
0929 11270                      CLS$M_NOOPEN_CIR         ; display.
0929 11271
0929 11272              $TRAN   TPA$_LAMBDA,TPA$_FAIL    ; Any other type is an error.
0929 11273
```

```
0929 11275 ;
0929 11276 ; REMOVE CONNECTIONS
0929 11277 ;
0929 11278         $STATE  REM_CON
0929 11279
0929 11280         $TRAN   '/',REM_CON_QUAL        ; Go to REM_CON_QUAL state if qualifier
0929 11281                                         ;  is present.
0929 11282
0929 11283         $TRAN   TPA$_LAMBDA,-           ; Otherwise remove all types of
0929 11284                 REM_NEXT_CLASS,-        ;  connections from the display.
0929 11285                 EXCLUDE_CLASS,,,-       ;
0929 11286                 CLS$M_ALL_CON           ;
0929 11287
0929 11288 ;
0929 11289 ; Process REMOVE CONNECTIONS qualifiers
0929 11290 ;
0929 11291         $STATE  REM_CON_QUAL
0929 11292
0929 11293         $TRAN   'TYPE',REM_CON_TYPE     ; REMOVE CONNECTIONS/TYPE
0929 11294         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929 11295
0929 11296 ;
0929 11297 ; Loop for multiple qualifiers on the CONNECTIONS class.
0929 11298 ;
0929 11299         $STATE NEXT_REM_CON_QUAL
0929 11300         $TRAN   '/',REM_CON_QUAL
0929 11301         $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929 11302
0929 11303         $STATE  REM_CON_TYPE
0929 11304         $TRAN   '='
0929 11305         $STATE
0929 11306         $TRAN   '(',REM_CON_TYPE_LOOP   ; Go parse a list of connection types.
0929 11307         $TRAN   TPA$_LAMBDA,            ; Go parse a single connection type.
0929 11308         $STATE
0929 11309         $TRAN   !REM_CON_TYPE_STRING,-  ;
0929 11310                 NEXT_REM_CON_QUAL       ;
0929 11311
0929 11312
0929 11313         $STATE  REM_CON_TYPE_LOOP       ; Parse a list of connection types till
0929 11314                                         ;  a ")" is found.
0929 11315         $TRAN   !REM_CON_TYPE_STRING    ;
0929 11316         $STATE
0929 11317         $TRAN   <',',>,REM_CON_TYPE_LOOP ; Go parse a single connection type.
0929 11318         $TRAN   ')',NEXT_REM_CON_QUAL   ; Exit loop if ")" found.
0929 11319
0929 11320
0929 11321         $STATE  REM_CON_TYPE_STRING     ; Parse a single connection type.
0929 11322
0929 11323         $TRAN   'ALL',TPA$_EXIT,-       ; "REM CONNECTION/TYPE=ALL" - Remove all
0929 11324                 EXCLUDE_CLASS,,,-       ;  types of connections from the
0929 11325                 CLS$M_ALL_CON           ;  display.
0929 11326
0929 11327         $TRAN   'OPEN',TPA$_EXIT,-      ; "REMOVE CONNECTIONS/OPEN" - Remove
0929 11328                 EXCLUDE_CLASS,,,-       ;  connections in the open state from
0929 11329                 CLS$M_OPEN_CON          ;  the display.
0929 11330
0929 11331         $TRAN   'NOOPEN',TPA$_EXIT,-    ; "REMOVE CONNECTIONS/NOOPEN" - Remove
```

```
0929 11332                      EXCLUDE_CLASS,,-         ;  connections in non-open states from
0929 11333                      CLS$M_NOOPEN_CON         ;  the display.
0929 11334
0929 11335              $TRAN   TPA$_LAMBDA,TPA$_FAIL    ; Any other type is an error.
0929 11336
0929 11337
```

```
0929 11339 ;
0929 11340 ; Set command.
0929 11341 ;
0929 11342
0929 11343          $STATE  SET
0929 11344          $TRAN   TPA$_EOS,TPA$_EXIT,-    ; Return if end of string.
0929 11345                  REFRESH
0929 11346
0929 11347          $TRAN   'INTERVAL',-            ; SET INTERVAL
0929 11348                  PARSE_INTERVAL          ;
0929 11349          $TRAN   'SCREEN',PARSE_SCREEN   ; SET SCREEN
0929 11350          $TRAN   TPA$_LAMBDA,SET_FIELD   ; Must be SET FIELD
0929 11351
0929 11352
0929 11353          $STATE  SET_FIELD
0929 11354          $TRAN   TPA$_SYMBOL,-           ; Check for valid field name.
0929 11355                  SET_FIELD_QUAL,-        ;
0929 11356                  LOOKUP_FIELD            ;
0929 11357
0929 11358          $STATE NEXT_SET_FIELD_QUAL
0929 11359          $TRAN   '/',SET_FIELD_QUAL      ; Loop on qualifiers to the fieldname.
0929 11360          $TRAN   TPA$_LAMBDA,SET_NEXT    ; Or else go on to next SET entity.
0929 11361
0929 11362          $STATE  SET_FIELD_QUAL
0929 11363          $TRAN   '/'                     ; Expect a qualifier.
0929 11364          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Else error.
0929 11365          $STATE
0929 11366          $TRAN   'WIDTH',SET_FIELD_WIDTH ; SET field-name/WIDTH.
0929 11367          $TRAN   'FORMAT',-              ; SET field-name/FORMAT.
0929 11368                  SET_FIELD_FORMAT        ;
0929 11369          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other qualifier is an error.
0929 11370
0929 11371          $STATE  SET_FIELD_WIDTH         ; SET field/WIDTH
0929 11372          $TRAN   !PARSE_VALUE,-          ; Get the value and set it.
0929 11373                  NEXT_SET_FIELD_QUAL,-   ;
0929 11374                  SET_WIDTH               ;
0929 11375
0929 11376          $STATE  SET_FIELD_FORMAT        ; Get the desired field format.
0929 11377          $TRAN   '='                     ; SET field/FORMAT=
0929 11378          $STATE                          ;
0929 11379          $TRAN   'DECIMAL',-             ; SET field/FORMAT=DECIMAL
0929 11380                  NEXT_SET_FIELD_QUAL,-   ;
0929 11381                  SET_FORMAT,,,DEC        ;
0929 11382          $TRAN   'HEXADECIMAL',-         ; SET field/FORMAT=HEXADECIMAL
0929 11383                  NEXT_SET_FIELD_QUAL,-   ;
0929 11384                  SET_FORMAT,,,HEX        ;
0929 11385
0929 11386          $STATE  SET_NEXT
0929 11387          $TRAN   <',">,SET               ; Loop on set command if comma follows
0929 11388                                          ;  the entity.
0929 11389          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; If not comma then redraw the header
0929 11390                  REFRESH                 ;  and return.
0929 11391
0929 11392          $STATE  PARSE_INTERVAL
0929 11393          $TRAN   !PARSE_VALUE,TPA$_EXIT,-; Get the value and set it.
0929 11394                  SET_INTERVAL            ;
0929 11395
```

```
0929 11396          $STATE  PARSE_SCREEN
0929 11397          $TRAN   !PARSE_VALUE,TPA$_EXIT,-; Get the value and set it.
0929 11398                  SET_SCREEN              ;
0929 11399
0929 11400          $STATE  PARSE_VALUE
0929 11401          $TRAN   '='                     ; Must begin with =
0929 11402          $STATE                          ;
0929 11403          $TRAN   <'-'>,TPA$_FAIL         ; Negative numbers are a no-no.
0929 11404          $TRAN   TPA$_DECIMAL,TPA$_EXIT,-; Store a decimal number at VALUE.
0929 11405                  ,VALUE                  ;
0929 11406          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; All else fails.
0929 11407
0929 11408
0929 11409 ;
0929 11410 ; Help command
0929 11411 ;
0929 11412
0929 11413          $STATE  HELP
0929 11414
0929 11415          $TRAN   TPA$_LAMBDA,TPA$_EXIT,GET_HELP
0929 11416
0929 11417          $END_STATE
0929 11418
0929 11419
```

```
                        0929 11421          .SBTTL  Tparse action routines.
                        0929 11422          .SBTTL          -       SHOW_CLUSTER
                        0929 11423 ;++
                        0929 11424 ;
                        0929 11425 ; FUNCTIONAL DESCRIPTION:
                        0929 11426 ;
                        0929 11427 ;       This routine sets the report to clusters.  It stores the parameters
                        0929 11428 ;       necessary for the FIND_ENTRY routine to locate a field descriptor for
                        0929 11429 ;       the cluster report.  It sets the bit in CURRENT_REPORT that indicates
                        0929 11430 ;       that the cluster report is being displayed.  It also saves a copy of
                        0929 11431 ;       the scroll flags for the local report and restores the scroll flags
                        0929 11432 ;       for the cluster report.
                        0929 11433 ;
                        0929 11434 ;
                        0929 11435 ; CALLING SEQUENCE:
                        0929 11436 ;
                        0929 11437 ;       CALLS   #0,SHOW_CLUSTER
                        0929 11438 ;
                        0929 11439 ; CALLED BY:
                        0929 11440 ;
                        0929 11441 ;       SWITCH_DISPLAY
                        0929 11442 ;       RESET
                        0929 11443 ;
                        0929 11444 ;
                        0929 11445 ; IMPLICIT INPUTS:
                        0929 11446 ;
                        0929 11447 ;       CLSTR_SCROLL_FLAGS
                        0929 11448 ;                       contains a mask of the scroll flags set from the last
                        0929 11449 ;                       time the cluster report was being displayed.
                        0929 11450 ;
                        0929 11451 ; IMPLICIT OUTPUTS:
                        0929 11452 ;
                        0929 11453 ;       ADRS_TABLE      contains the address of the cluster address table.
                        0929 11454 ;       BITMAP_ADRS     contains the address of the cluster bitmap.
                        0929 11455 ;       KEYTAB_ADRS     contains the address of the cluster keyword table.
                        0929 11456 ;       MAP_LEN         contains the the length of the cluster bitmap.
                        0929 11457 ;       CURRENT_REPORT  Set to clusters by setting RPT$M_REPORT.
                        0929 11458 ;       SCROLL_FLAGS    restored to from the last time the cluster report was
                        0929 11459 ;                       displayed.
                        0929 11460 ;
                        0929 11461 ; RETURN STATUS:
                        0929 11462 ;
                        0929 11463 ;       SS$_NORMAL or failed status returned in R0.
                        0929 11464 ;
                        0929 11465 ;--
                        0929 11466
                  0000  0929 11467 .ENTRY  SHOW_CLUSTER,^M<>
                        092B 11468
              1AA2'  30 092B 11469          BSBW    CLEAR_SCREEN            ; Clear the screen.
            6D 50    E9 092E 11470          BLBC    R0,99$                  ; Branch on error.
     0000055B'EF     7F 0931 11471          PUSHAQ  SCREEN_DSCR             ; Set up a buffer for the screen.
00000000'GF  01     FB 0937 11472          CALLS   #1,G^LIB$SET_BUFFER     ;
            5D 50    E9 093E 11473          BLBC    R0,99$                  ; Branch on error.
               00    E2 0941 11474          BBSS    #RPT$V_CLSTR,-          ; Branch if we're already displaying
         50 F6BA CF      0943 11475                  CURRENT_REPORT,10$      ;  the cluster report.  Otherwise set
                         0947 11476                                          ;  the current report to clusters.
     0000047A'EF     90 0947 11477          MOVB    SCROLL_FLAGS,-          ; Store the scroll flags for the local
```

```
                0000047B'EF          094D 11478                 LOCAL_SCROLL_FLAGS      ;  report.
                0000047C'EF    90    0952 11479         MOVB    CLSTR_SCROLL_FLAGS,-    ; Get the cluster report scroll flags.
                0000047A'EF          0958 11480                 SCROLL_FLAGS            ;
      00002200'EF    F6A0 CF    D0    095D 11481         MOVL    CLASS_SELECT,LOCAL_SAVE ; Save local report class selections.
      F692 CF   00002204'EF    D0    0966 11482         MOVL    CLUST_SAVE,CLASS_SELECT ; Get cluster report class selections.
                00000000'EF    DE    096F 11483         MOVAL   CLUSTER_KEYTAB,-        ; Store address of cluster keyword
                00002274'EF          0975 11484                 KEYTAB_ADRS             ;  table.
  00002264'EF   00000000'EF    DE    097A 11485         MOVAL   CLUSTER_TABLE,ADRS_TABLE; Store address of cluster address
                                     0985 11486                                         ;  table.
                00000000'EF    DE    0985 11487         MOVAL   CLUSTER_BITMAP,-        ; Store address of cluster bitmap.
                00002268'EF          098B 11488                 BITMAP_ADRS             ;
          0000226C'EF    02    D0    0990 11489         MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Store length of cluster bitmap.
          00002348'EF    00    FB    0997 11490 10$:    CALLS   #0,REFRESH              ; Redraw the header.
                               04    099E 11491 99$:    RET
                                     099F 11492
```

```
                        099F 11494          .SBTTL          -       SHOW_LOCAL
                        099F 11495 ;++
                        099F 11496 ;
                        099F 11497 ; FUNCTIONAL DESCRIPTION:
                        099F 11498 ;
                        099F 11499 ;       This routine sets the report to local ports.  It stores the parameters
                        099F 11500 ;       necessary for the FIND_ENTRY routine to locate a field descriptor for
                        099F 11501 ;       the local report.  It sets the bit in CURRENT_REPORT that indicates
                        099F 11502 ;       that the local report is being displayed.  It also saves a copy of
                        099F 11503 ;       the scroll flags for the cluster report and restores the scroll flags
                        099F 11504 ;       for the local report.
                        099F 11505 ;
                        099F 11506 ; CALLING SEQUENCE:
                        099F 11507 ;
                        099F 11508 ;       CALLS   #0,SHOW_LOCAL
                        099F 11509 ;
                        099F 11510 ; CALLED BY:
                        099F 11511 ;
                        099F 11512 ;       SWITCH_DISPLAY
                        099F 11513 ;
                        099F 11514 ; IMPLICIT INPUTS:
                        099F 11515 ;
                        099F 11516 ;       LOCAL_SCROLL_FLAGS
                        099F 11517 ;                       contains a mask of the scroll flags set from the last
                        099F 11518 ;                       time the cluster report was being displayed.
                        099F 11519 ;
                        099F 11520 ; IMPLICIT OUTPUTS:
                        099F 11521 ;
                        099F 11522 ;       ADRS_TABLE      contains the address of the local status address table.
                        099F 11523 ;       BITMAP_ADRS     contains the address of the local status bitmap.
                        099F 11524 ;       KEYTAB_ADRS     contains the address of the local status keyword table.
                        099F 11525 ;       MAP_LEN         contains the the length of the local status bitmap.
                        099F 11526 ;       CURRENT_REPORT  Set to local report by clearing RPT$M_REPORT.
                        099F 11527 ;       SCROLL_FLAGS    restored to from the last time the cluster report was
                        099F 11528 ;                       displayed.
                        099F 11529 ;
                        099F 11530 ; RETURN STATUS:
                        099F 11531 ;
                        099F 11532 ;       SS$_NORMAL or failed status returned in R0.
                        099F 11533 ;
                        099F 11534 ;--
                        099F 11535
                   0000 099F 11536 .ENTRY  SHOW_LOCAL,^M<>
                        09A1 11537
               1A2C'  30 09A1 11538          BSBW    CLEAR_SCREEN                ; Clear the screen.
             6D 50    E9 09A4 11539          BLBC    R0,99$                      ; Branch on error.
      0000055B'EF     7F 09A7 11540          PUSHAQ  SCREEN_DSCR                 ; Set up a buffer for the screen.
00000000'GF    01     FB 09AD 11541          CALLS   #1,G^LIB$SET_BUFFER         ;
             5D 50    E9 09B4 11542          BLBC    R0,99$                      ; Branch on error.
                00    E5 09B7 11543          BBCC    #RPT$V_CLSTR,-              ; Branch if we are already displaying
          50 F644 CF      09B9 11544                  CURRENT_REPORT,10$          ;  local report.  Otherwise, set the
                        09BD 11545                                              ;  current report to local.
      0000047A'EF     90 09BD 11546          MOVB    SCROLL_FLAGS,-              ; Store the current scroll flags for
      0000047C'EF        09C3 11547                  CLSTR_SCROLL_FLAGS          ;  the cluster report.
      0000047B'EF     90 09C8 11548          MOVB    LOCAL_SCROLL_FLAGS,-        ; Get the local report scroll flags.
      0000047A'EF        09CE 11549                  SCROLL_FLAGS                ;
00002204'EF   F62A CF   D0 09D3 11550          MOVL    CLASS_SELECT,CLUST_SAVE     ; Save cluster report class selections.
```

```
                F61C CF   00002200'EF   D0  09DC 11551          MOVL    LOCAL_SAVE,CLASS_SELECT ; Get local report class selections.
00002274'EF     00000000'EF   DE  09E5 11552          MOVAL   LOCAL_KEYTAB,KEYTAB_ADRS; Store address of local status keyword
                                  09F0 11553                                          ;  table.
00002264'EF     00000000'EF   DE  09F0 11554          MOVAL   LOCAL_TABLE,ADRS_TABLE  ; Store address of local table.
00002268'EF     00000000'EF   DE  09FB 11555          MOVAL   LOCAL_BITMAP,BITMAP_ADRS; Store address of local bitmap.
                0000226C'EF   00  D0  0A06 11556          MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Store length of local bitmap.
                00002348'EF   00  FB  0A0D 11557 10$:     CALLS   #0,REFRESH              ; Redraw the header.
                                  04  0A14 11558 99$:     RET
                                      0A15 11559
                                      0A15 11560
```

```
                                .SBTTL          -       ADD_FIELD
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine is used to add a field to the display by setting the bit
                        ;       corresponding to the field in the bitmap for the current report.  If
                        ;       the field is in a class which is currently not being displayed, then
                        ;       that class is enabled along with the desired field, but all other
                        ;       fields in that class are turned off.  If the class being enabled is
                        ;       the Cluster class, then update FIRST_DATA_LINE so that the scrolling
                        ;       region does not encompass the Cluster class display.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       CALLS   #0,ADD_FIELD
                        ;
                        ; CALLED_BY:
                        ;
                        ;       READ_W_PROMPT
                        ;
                        ; INPUTS:
                        ;
                        ;       KEY_OFF = Address of longword in the bitmap that contains the bit
                        ;                 corresponding to the field selected.
                        ;       KEY_MSK = Mask of the bit within KEY_OFF corresponding to the field.
                        ;
                        ; OUTPUTS:
                        ;
                        ;       Bit corresponding to the field set in the bitmap.
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       SS$_NORMAL or failed status returned in R0.
                        ;
                        ; REGISTER USAGE:
                        ;
                        ;       R2 -    Points to the address containing the bit corresponding to the
                        ;               field.
                        ;
                        ;--
                        ;
                 01FC   .ENTRY  ADD_FIELD,^M<R2,R3,R4,R5,R6,R7,R8>
00000B3A'EF    00   FB  CALLS   #0,LOOKUP_FIELD         ; Check for valid field name.
                        JLBC    R0,99$                  ; Branch on error.
52  00000543'EF   D0    MOVL    KEY_OFF,R2              ; Get address of longword containing
                                                        ;  the bit.
53  00000548'EF   D0    MOVL    KEY_DSCR,R3             ; Get the address of the descriptor for
                                                        ;  the field.
          0C A3   D3    BITL    FD$L_CLASS(R3),-        ; Is the class for this field already
              F5C9 CF           CLASS_SELECT            ;  enabled?
                        JNEQ    70$                     ; Branch if so.
          03    E1    BBC     #CLS$V_CIR,-            ; Is this field in the circuits class?
             0B 0C A3           FD$L_CLASS(R3),10$      ;
      00000608 8F   C8  BISL    #CLS$M_ALL_CIR,-        ; If so, enable all types of circuits.
              F5B6 CF           CLASS_SELECT            ;
          22    11    BRB     30$                     ; Jump to common code.
```

```
                     04   E1  0A4D 11619 10$:    BBC     #CLS$V_CON,-            ; Is this field in the connections
                0B 0C A3        0A4F 11620              FD$L_CLASS(R3),20$      ;  class?
            00001810 8F   C8    0A52 11621          BISL    #CLS$M_ALL_CON,-        ; If so, enable all types of
                 F5A6 CF        0A58 11622              CLASS_SELECT            ;  connections.
                     12   11    0A5B 11623          BRB     30$                     ; Jump to common code.
                     00   E1    0A5D 11624 20$:     BBC     #CLS$V_CLUB,-           ; Is this field in the cluster class?
                07 0C A3        0A5F 11625              FD$L_CLASS(R3),25$      ;
      00002254'EF    0E   D0    0A62 11626          MOVL    #14,FIRST_DATA_LINE     ; If so, set first line to be scrolled.
                   0C A3   C8    0A69 11627 25$:     BISL    FD$L_CLASS(R3),-        ; Enable the class.
                F592 CF         0A6C 11628              CLASS_SELECT            ;
   58   00002268'EF    D0    0A6F 11629 30$:     MOVL    BITMAP_ADRS,R8          ; Get the address of field descriptor
                               0A76 11630                                      ;  bitmap for the current report.
        00002270'EF    D4    0A76 11631          CLRL    START_BIT               ; Set up for FIND_ENTRY.
        0000046C'EF    D4    0A7C 11632          CLRL    H_SCROLL_LINE           ; Make sure that fields scrolled off
                               0A82 11633                                      ;  the screen are also checked.
                     00   E1    0A82 11634          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
                09 F579 CF      0A84 11635              CURRENT_REPORT,50$      ;  cluster report.
      0000226C'EF    02   D0    0A88 11636          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Save length of cluster bitmap.
                     07   11    0A8F 11637          BRB     60$                     ; Jump to common code.
      0000226C'EF    00   D0    0A91 11638 50$:     MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Save length of local bitmap.
                  18C8'   30    0A98 11639 60$:     BSBW    FIND_ENTRY              ; Search for any field being displayed.
                   1D 50  E9    0A9B 11640          BLBC    R0,70$                  ; Branch if none found.
                   0C AA  D1    0A9E 11641          CMPL    FD$L_CLASS(R10),-       ; Is the class for the field we just
                   0C A3        0AA1 11642              FD$L_CLASS(R3)          ;  found the same as the field we are
                               0AA3 11643                                      ;  adding to the display?
                     F3   12    0AA3 11644          BNEQ    60$                     ; If not go find another field.
   54   00002270'EF    01   C3    0AA5 11645          SUBL3   #1,START_BIT,R4         ; Get the number of the bit in the
                               0AAD 11646                                      ;  bitmap that corresponds to the field
                               0AAD 11647                                      ;  we just found.
                     55   D4    0AAD 11648          CLRL    R5                      ; Set up for extended divide.
57   56   54   08   7B    0AAF 11649          EDIV    #8,R4,R6,R7             ; R6 gets the index into the address
                               0AB4 11650                                      ;  table.  R7 gets the bit position.
     DF 6846    57   E5    0AB4 11651          BBCC    R7,(R8)[R6],60$         ; Disable the field and loop.
                     DD   11    0AB9 11652          BRB     60$                     ; Loop for all field descriptors.
 62    00000547'EF    C8    0ABB 11653 70$:     BISL    KEY_MSK,(R2)            ; Set the bit, turn on the field.
           50   01   D0    0AC2 11654          MOVL    #SS$_NORMAL,R0
                     04   0AC5 11655 99$:     RET
                               0AC6 11656
                               0AC6 11657
```

```
                     0AC6 11659          .SBTTL          -       REMOVE_FIELD
                     0AC6 11660 ;++
                     0AC6 11661 ;
                     0AC6 11662 ; FUNCTIONAL DESCRIPTION:
                     0AC6 11663 ;
                     0AC6 11664 ;       This routine is used to remove a field from the display by clearing the
                     0AC6 11665 ;       bit corresponding to the field in the bitmap for the current report.
                     0AC6 11666 ;       If the field being cleared is the last field being displayed in a
                     0AC6 11667 ;       particular class, then the class is turned off instead.  If the class
                     0AC6 11668 ;       being turned off is the Cluster class, then update FIRST_DATA_LINE so
                     0AC6 11669 ;       that the scrolling region uses the full display.
                     0AC6 11670 ;
                     0AC6 11671 ; CALLING SEQUENCE:
                     0AC6 11672 ;
                     0AC6 11673 ;       CALLS   #0,REMOVE_FIELD
                     0AC6 11674 ;
                     0AC6 11675 ; CALLED_BY:
                     0AC6 11676 ;
                     0AC6 11677 ;       READ_W_PROMPT
                     0AC6 11678 ;
                     0AC6 11679 ; INPUTS:
                     0AC6 11680 ;
                     0AC6 11681 ;       KEY_OFF = Address of longword in the bitmap that contains the bit
                     0AC6 11682 ;                 corresponding to the field selected.
                     0AC6 11683 ;       KEY_MSK = Mask of the bit within KEY_OFF corresponding to the field.
                     0AC6 11684 ;
                     0AC6 11685 ; OUTPUTS:
                     0AC6 11686 ;
                     0AC6 11687 ;       Bit corresponding to the field cleared in the bitmap, or bit
                     0AC6 11688 ;       corresponding to the class cleared at CLASS_SELECT.
                     0AC6 11689 ;
                     0AC6 11690 ; RETURN STATUS:
                     0AC6 11691 ;
                     0AC6 11692 ;       SS$_NORMAL or failed status returned in R0.
                     0AC6 11693 ;
                     0AC6 11694 ; REGISTER USAGE:
                     0AC6 11695 ;
                     0AC6 11696 ;       R2 -    Points to the address containing the bit corresponding to the
                     0AC6 11697 ;               field.
                     0AC6 11698 ;
                     0AC6 11699 ;--
                 010C 0AC6 11700 .ENTRY  REMOVE_FIELD,^M<R2,R3,R8>
00000B3A'EF    00    FB 0AC8 11701          CALLS   #0,LOOKUP_FIELD         ; Check for valid field name.
          67 50      E9 0ACF 11702          BLBC    R0,99$                  ; Branch on error.
52  00000543'EF      D0 0AD2 11703          MOVL    KEY_OFF,R2              ; Get address of longword containing
                     0AD9 11704                                             ;  the bit.
53  0000054B'EF      D0 0AD9 11705          MOVL    KEY_DSCR,R3             ; Get the address of the descriptor
                     0AE0 11706                                             ;  for the field.
58  00002268'EF      D0 0AE0 11707          MOVL    BITMAP_ADRS,R8          ; Get the address of the field
                     0AE7 11708                                             ;  descriptor bitmap for the current
                     0AE7 11709                                             ;  report.
    00002270'EF      D4 0AE7 11710          CLRL    START_BIT               ; Set up for FIND_ENTRY.
    0000046C'EF      D4 0AED 11711          CLRL    H_SCROLL_LINE           ; Make sure that fields scrolled off
                     0AF3 11712                                             ;  the screen are also checked.
              00     E1 0AF3 11713          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
        09 F508 CF      0AF5 11714                  CURRENT_REPORT,20$      ;  cluster report.
0000226C'EF    02    D0 0AF9 11715          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Save length of cluster bitmap.
```

```
                    07   11   0B00 11716                BRB     30$                             ; Jump to common code.
0000226C'EF         00   D0   0B02 11717 20$:          MOVL    #LOCAL_MAP_LEN,MAP_LEN          ; Save length of local bitmap.
                  1857'  30   0B09 11718 30$:          BSBW    FIND_ENTRY                      ; Search for any field being displayed.
                15 50    E9   0B0C 11719                BLBC    R0,40$                          ; Branch if none found.
                0C AA    D1   0B0F 11720                CMPL    FD$L_CLASS(R10),-               ; Is the class for the field we just
                0C A3         0B12 11721                        FD$L_CLASS(R3)                  ;  found the same as the field we are
                              0B14 11722                                                        ;  adding to the display?
                   F3   12   0B14 11723                BNEQ    30$                             ; If not go find another field.
              5A   53   D1   0B16 11724                CMPL    R3,R10                          ; Is it the same field?
                   EE   13   0B19 11725                BEQL    30$                             ; If so go find another field.
62    00000547'EF    CA   0B1B 11726                BICL    KEY_MSK,(R2)                    ; Clear the bit, turn off the field.
                   15   11   0B22 11727                BRB     99$                             ; Branch to common code.
                0C A3    CA   0B24 11728 40$:          BICL    FD$L_CLASS(R3),-                ; No other fields in this class found,
              F4D7 CF         0B27 11729                        CLASS_SELECT                    ;  so turn off the class, not the field.
                   00   E1   0B2A 11730                BBC     #CLS$V_CLUB,-                   ; Did we just turn off the cluster
             07 0C A3         0B2C 11731                        FD$L_CLASS(R3),50$              ;  class?
00002254'EF         07   D0   0B2F 11732                MOVL    #7,FIRST_DATA_LINE              ; If so, set first line to be scrolled.
                50   01   D0   0B36 11733 50$:          MOVL    #SS$_NORMAL,R0                  ; Return success.
                          04   0B39 11734 99$:          RET
                               0B3A 11735
                               0B3A 11736
```

```
                     0B3A 11738          .SBTTL          -       LOOKUP_FIELD
                     0B3A 11739 ;++
                     0B3A 11740 ;
                     0B3A 11741 ; FUNCTIONAL DESCRIPTION:
                     0B3A 11742 ;
                     0B3A 11743 ;        This routine is used to parse the field name by calling
                     0B3A 11744 ;       LIB$LOOKUP_KEY using each of the keyword tables containing the field
                     0B3A 11745 ;       names.  The keyword value is converted into an address and mask for the
                     0B3A 11746 ;       bit within the appropriate bitmap and the address of the descriptor for
                     0B3A 11747 ;       the field.
                     0B3A 11748 ;
                     0B3A 11749 ; CALLING SEQUENCE:
                     0B3A 11750 ;
                     0B3A 11751 ;       CALLS   #0,LOOKUP_FIELD
                     0B3A 11752 ;
                     0B3A 11753 ; CALLED BY:
                     0B3A 11754 ;
                     0B3A 11755 ;       ADD_FIELD
                     0B3A 11756 ;       REMOVE_FIELD
                     0B3A 11757 ;
                     0B3A 11758 ; INPUTS:
                     0B3A 11759 ;
                     0B3A 11760 ; IMPLICIT INPUTS:
                     0B3A 11761 ;
                     0B3A 11762 ;       Selected fieldname stored at address TPARSE_BLK+TPA$L_TOKENCNT.
                     0B3A 11763 ;       Address of keyword table stored at KEYTAB_ADRS.
                     0B3A 11764 ;       Address of bitmap stored at BITMAP_ADRS.
                     0B3A 11765 ;       Address of address table stored at ADRS_TABLE.
                     0B3A 11766 ;
                     0B3A 11767 ; OUTPUTS:
                     0B3A 11768 ;
                     0B3A 11769 ;       KEY_OFF  = Address of longword in the bitmap that contains the bit
                     0B3A 11770 ;                  corresponding to the field selected.
                     0B3A 11771 ;       KEY_MSK  = Mask of the bit within KEY_OFF corresponding to the field.
                     0B3A 11772 ;       KEY_DSCR = Address of the field descriptor for the selected field.
                     0B3A 11773 ;
                     0B3A 11774 ; RETURN STATUS:
                     0B3A 11775 ;
                     0B3A 11776 ;       SS$_NORMAL or failed status from LIB$LOOKUP_KEY returned in R0.
                     0B3A 11777 ;
                     0B3A 11778 ;--
                     0B3A 11779
                003C 0B3A 11780 .ENTRY  LOOKUP_FIELD,^M<R2,R3,R4,R5>
53   00002274'EF   D0 0B3C 11781         MOVL    KEYTAB_ADRS,R3          ; Get address of keyword table.
54   00002268'EF   D0 0B43 11782         MOVL    BITMAP_ADRS,R4          ; Get address of bitmap.
55   00002264'EF   D0 0B4A 11783         MOVL    ADRS_TABLE,R5           ; Get address of address table.
     0000053F'EF   DF 0B51 11784         PUSHAL  KEY_VAL                 ; Push address to receive value.
           53      DD 0B57 11785         PUSHL   R3                      ; Push keyword table address.
                      0B59 11786                                         ; Push address of descriptor for the
                      0B59 11787                                         ;  remainder of the string.
     000004D7'EF   DF 0B59 11788         PUSHAL  TPARSE_BLK+TPA$L_TOKENCNT
00000000'GF   03   FB 0B5F 11789         CALLS   #3,G^LIB$LOOKUP_KEY     ; Check for valid field name in the
                      0B66 11790                                         ;  current report.
           2D 50   E8 0B66 11791         BLBS    R0,30$                  ; Branch if field name is valid in the
                      0B69 11792                                         ;  current report.
                      0B69 11793 ;
                      0B69 11794 ; If the field lookup fails, then determine if it is a valid field name,
```

```
                                0B69 11795 ; but simply can't be displayed in the current report.  If so, then pass
                                0B69 11796 ; the additional information on to the user.
                                0B69 11797 ;
                          00    E1  0B69 11798          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
                    08 F492 CF        0B6B 11799                  CURRENT_REPORT,10$      ;  cluster report.
                00000000'EF       DF  0B6F 11800          PUSHAL  LOCAL_KEYTAB            ; Lookup field in the LOCAL keyword
                                0B75 11801                                              ;  table.
                          06    11  0B75 11802          BRB     20$                     ; Jump to common code.
                00000000'EF       DF  0B77 11803 10$:     PUSHAL  CLUSTER_KEYTAB          ; Check for valid field name.
                000004D7'EF       DF  0B7D 11804 20$:     PUSHAL  TPARSE_BLK+TPA$L_TOKENCNT
          00000000'GF    02       FB  0B83 11805          CALLS   #2,G^LIB$LOOKUP_KEY     ; Check for valid field name.
                    3C  50       E9  0B8A 11806          BLBC    R0,99$                  ; Branch on error.
          50    00000000'8F       D0  0B8D 11807          MOVL    #SHOW$_NOAVAIL,R0       ; "Field not available in this report".
                          33    11  0B94 11808          BRB     99$                     ; Exit with status.
          52    0000053F'EF       D0  0B96 11809 30$:     MOVL    KEY_VAL,R2              ; Get the keyword value in R2,R3.
                          53    D4  0B9D 11810          CLRL    R3                      ;
         53   52   52    20    7B  0B9F 11811          EDIV    #32,R2,R2,R3            ; Convert the keyword value into an
                                0BA4 11812                                              ;  offset into a field bitmap.
                                0BA4 11813                                              ;  R2 gets the longword offset,
                                0BA4 11814                                              ;  R3 gets the bit offset.
        00000543'EF    6442    DE  0BA4 11815          MOVAL   (R4)[R2],KEY_OFF        ; Get address of longword containing
                                0BAC 11816                                              ;  the bit and store in KEY_OFF.
                                0BAC 11817 ;
                                0BAC 11818 ; Convert the bit position in R3 to a mask.
                                0BAC 11819 ;
                    54    01    D0  0BAC 11820          MOVL    #1,R4                   ; Start with mask for bit 0.
      00000547'EF   54    53    78  0BAF 11821          ASHL    R3,R4,KEY_MSK           ; Shift left by the number a bits in R3
                                0BB7 11822                                              ;  and save in KEY_MSK.
                                0BB7 11823
          54    0000053F'EF       D0  0BB7 11824          MOVL    KEY_VAL,R4
      0000054B'EF    6544       D0  0BBE 11825          MOVL    (R5)[R4],KEY_DSCR       ; Get address of field descriptor for
                                0BC6 11826                                              ;  this keyword and save in KEY_DSCR.
                    50    01    D0  0BC6 11827          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                          04    0BC9 11828 99$:         RET
                                0BCA 11829
                                0BCA 11830
```

```
                     0BCA 11832          .SBTTL          -       ADD_ALL_FIELDS
                     0BCA 11833 ;++
                     0BCA 11834 ;
                     0BCA 11835 ; FUNCTIONAL DESCRIPTION:
                     0BCA 11836 ;
                     0BCA 11837 ;       This routine is used to turn on all the fields in a particular class
                     0BCA 11838 ;       when the /ALL qualifier is used with the ADD class-name command.
                     0BCA 11839 ;
                     0BCA 11840 ; CALLING SEQUENCE:
                     0BCA 11841 ;
                     0BCA 11842 ;       BRB     ADD_ALL_FIELDS
                     0BCA 11843 ;
                     0BCA 11844 ; CALLED BY:
                     0BCA 11845 ;
                     0BCA 11846 ;       READ_W_PROMPT
                     0BCA 11847 ;
                     0BCA 11848 ; INPUTS:
                     0BCA 11849 ;
                     0BCA 11850 ;       Class selector bit for the class to have all fields turned on at
                     0BCA 11851 ;       TPARSE_BLK+TPA$L_PARAM.
                     0BCA 11852 ;
                     0BCA 11853 ; OUTPUTS:
                     0BCA 11854 ;
                     0BCA 11855 ;       Bits corresponding to the the fields in the selected class set in the
                     0BCA 11856 ;       bitmap.
                     0BCA 11857 ;       Bit corresponding to the selected class set at CLASS_SELECT.
                     0BCA 11858 ;
                     0BCA 11859 ;
                     0BCA 11860 ;--
                     0BCA 11861
             06E0    0BCA 11862 .ENTRY  ADD_ALL_FIELDS,^M<R5,R6,R7,R9,R10>
00000C2D'EF    00 FB 0BCC 11863          CALLS   #0,INCLUDE_CLASS        ; Make sure the class is add to the
                     0BD3 11864                                          ;  display.
56  00002264'EF    D0 0BD3 11865          MOVL    ADRS_TABLE,R6           ; Get the address of the address table
                     0BDA 11866                                          ;  for the field descriptors.
57  00002268'EF    D0 0BDA 11867          MOVL    BITMAP_ADRS,R7          ; Get the address of field descriptor
                     0BE1 11868                                          ;  bitmap for the current report.
        59    01    D0 0BE1 11869          MOVL    #1,R9                   ; Start with bit 0.
              00    E1 0BE4 11870          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
     09 F417  CF       0BE6 11871                  CURRENT_REPORT,10$      ;  cluster report.
55  00000000'EF    D0 0BEA 11872          MOVL    CLUSTER_KEYTAB,R5       ; Get the number of cluster keyword
                     0BF1 11873                                          ;  vectors.
              07    11 0BF1 11874          BRB     15$                     ; Jump to common code.
55  00000000'EF    D0 0BF3 11875 10$:     MOVL    LOCAL_KEYTAB,R5         ; Get the number of local keyword
                     0BFA 11876                                          ;  vectors.
 55   55    FF 8F 78 0BFA 11877 15$:     ASHL    #-1,R5,R5               ; Get the number of fields to check.
              5A    D4 0BFF 11878 16$:     CLRL    R10                     ; Dividend must be a quadword.
5A   51    59    20 7B 0C01 11879          EDIV    #32,R9,R1,R10           ; Calculate the number of bits left to
     5A    20    5A C3 0C06 11880          SUBL3   R10,#32,R10             ;  scan in the longword.
59   67    5A    59 EB 0C0A 11881          FFC     R9,R10,(R7),R9          ; Scan for a clear bit.
              EE    13 0C0F 11882          BEQLU   16$                     ; If not, scan next longword.
        55    59    D1 0C11 11883          CMPL    R9,R5                   ; Have we checked all fields?
              16    13 0C14 11884          BEQL    99$                     ; Branch if so.
                     0C16 11885
        5A   6649   D0 0C16 11886 20$:     MOVL    (R6)[R9],R10            ; Get the address of the field
                     0C1A 11887                                          ;  descriptor.
           0C AA    D3 0C1A 11888          BITL    FD$L_CLASS(R10),-       ; Is the field in the desired class?
```

```
000004E7'EF          0C1D 11889                 TPARSE_BLK+TPA$L_PARAM  ;
                04   0C22 11890          BEQL   25$                     ; If not go find another field.
 00 67          59   E2   0C24 11891      BBSS   R9,(R7),25$             ; Otherwise, turn on the field.
                59   D6   0C28 11892 25$: INCL   R9                      ; Get number of next bit to test.
                D3   11   0C2A 11893      BRB    16$                     ; Loop for all fields.
                04   0C2C 11894 99$:     RET
                     0C2D 11895
                     0C2D 11896
```

```
                        0C2D 11898          .SBTTL          -       INCLUDE_CLASS
                        0C2D 11899 ;++
                        0C2D 11900 ;
                        0C2D 11901 ; FUNCTIONAL DESCRIPTION:
                        0C2D 11902 ;
                        0C2D 11903 ;       This routine is used to add a class to the display.  It simply sets the
                        0C2D 11904 ;       bit that corresponds to the desired class in location CLASS_SELECT.
                        0C2D 11905 ;
                        0C2D 11906 ; CALLING SEQUENCE:
                        0C2D 11907 ;
                        0C2D 11908 ;       CALLS   #0,INCLUDE_CLASS
                        0C2D 11909 ;
                        0C2D 11910 ; CALLED BY:
                        0C2D 11911 ;
                        0C2D 11912 ;       READ_W_PROMPT
                        0C2D 11913 ;       ADD_ALL_FIELDS
                        0C2D 11914 ;
                        0C2D 11915 ; INPUTS:
                        0C2D 11916 ;
                        0C2D 11917 ;       Mask of the bit to be set at address TPARSE_BLK+TPA$L_PARAM.
                        0C2D 11918 ;
                        0C2D 11919 ; OUTPUTS:
                        0C2D 11920 ;
                        0C2D 11921 ;       Bit corresponding to the desired class set at CLASS_SELECT.
                        0C2D 11922 ;
                        0C2D 11923 ; RETURN STATUS:
                        0C2D 11924 ;
                        0C2D 11925 ;       SS$_NORMAL if class can be added to the current report, otherwise
                        0C2D 11926 ;       SHOW$_CACLS if the class can't be added to the current report.
                        0C2D 11927 ;
                        0C2D 11928 ;--
                        0C2D 11929
                0000    0C2D 11930 .ENTRY  INCLUDE_CLASS,^M<>
                        0C2F 11931
                   00 E1 0C2F 11932          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
                 OF F3CC CF    0C31 11933          CURRENT_REPORT,10$      ;  cluster report.
                00000180 8F D3 0C35 11934          BITL    #CLS$M_PRT!CLS$M_ERR,- ; Is the desired class in the
                000004E7'EF    0C3B 11935          TPARSE_BLK+TPA$L_PARAM  ;  LOCAL_PORTS report?
                        2C 12  0C40 11936          BNEQ    98$             ; Error if so.
                        0D 11  0C42 11937          BRB     20$             ; Branch to common code.
                               0C44 11938
                0000007F 8F D3 0C44 11939 10$:     BITL    #CLS$M_ANY_CLUS,-       ; Is the desired class in the CLUSTER
                000004E7'EF    0C4A 11940          TPARSE_BLK+TPA$L_PARAM  ;  report?
                        1D 12  0C4F 11941          BNEQ    98$             ; Error if so.
                               0C51 11942
                000004E7'EF C8 0C51 11943 20$:     BISL    TPARSE_BLK+TPA$L_PARAM,-; Set the bit corresponding to the
                     F3A7 CF   0C57 11944          CLASS_SELECT            ;  desired class.
                           00 E1 0C5A 11945          BBC     #CLS$V_CLUB,-           ; Branch if desired class isn't
                000004E7'EF    0C5C 11946          TPARSE_BLK+TPA$L_PARAM,-;  CLUSTER
                           07  0C61 11947          30$
            00002254'EF    0E D0 0C62 11948          MOVL    #14,FIRST_DATA_LINE     ; Set first line to be scrolled.
                        50 01 D0 0C69 11949 30$:     MOVL    #SS$_NORMAL,R0          ; Indicate success.
                        12 11  0C6C 11950          BRB     99$             ; Branch to common exit.
                               0C6E 11951
000004A9'EF     00000495'EF    DE 0C6E 11952 98$:     MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for class name.
             50 00000000'8F    D0 0C79 11953          MOVL    #SHOW$_CACLS,R0         ; "Can't add the class to this report"
                               04 0C80 11954 99$:     RET
```

```
0C81 11955
0C81 11956
```

```
                        0C81 11958          .SBTTL          -       EXCLUDE_CLASS
                        0C81 11959 ;++
                        0C81 11960 ;
                        0C81 11961 ; FUNCTIONAL DESCRIPTION:
                        0C81 11962 ;
                        0C81 11963 ;       This routine is used to remove a class from the display.  It simply
                        0C81 11964 ;       clears the bit that corresponds to the desired class in location
                        0C81 11965 ;       CLASS_SELECT.
                        0C81 11966 ;
                        0C81 11967 ; CALLING SEQUENCE:
                        0C81 11968 ;
                        0C81 11969 ;       CALLS   #0,EXCLUDE_CLASS
                        0C81 11970 ;
                        0C81 11971 ; CALLED BY:
                        0C81 11972 ;
                        0C81 11973 ;       READ_W_PROMPT
                        0C81 11974 ;
                        0C81 11975 ; INPUTS:
                        0C81 11976 ;
                        0C81 11977 ;       Mask of the bit to be cleared at address TPARSE_BLK+TPA$L_PARAM.
                        0C81 11978 ;
                        0C81 11979 ; OUTPUTS:
                        0C81 11980 ;
                        0C81 11981 ;       Bit corresponding to the desired class cleared at CLASS_SELECT.
                        0C81 11982 ;
                        0C81 11983 ; RETURN STATUS:
                        0C81 11984 ;
                        0C81 11985 ;       SS$_NORMAL if class can be removed from the current report, otherwise
                        0C81 11986 ;       SHO$_CRCLS if the class can't be removrd from the current report.
                        0C81 11987 ;
                        0C81 11988 ;--
                        0C81 11989
                0000    0C81 11990 .ENTRY  EXCLUDE_CLASS,^M<>
                        0C83 11991
                   00   0C83 11992          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
         0F F378 CF     0C85 11993                  CURRENT_REPORT,10$      ;  cluster report.
         00000180 8F    0C89 11994          BITL    #CLS$M_CPRT!CLS$M_ERR,- ; Is the desired class in the
         000004E7'EF    0C8F 11995                  TPARSE_BLK+TPA$L_PARAM  ;  LOCAL_PORTS report?
                   2C   0C94 11996          BNEQ    98$                     ; Error if so.
                   0D   0C96 11997          BRB     20$                     ; Branch to common code.
                        0C98 11998
         0000007F 8F    0C98 11999 10$:     BITL    #CLS$M_ANY_CLUS,-       ; Is the desired class in the CLUSTER
         000004E7'EF    0C9E 12000                  TPARSE_BLK+TPA$L_PARAM  ;  report?
                   1D   0CA3 12001          BNEQ    98$                     ; Error if so.
                        0CA5 12002
         000004E7'EF    CA   0CA5 12003 20$:     BICL    TPARSE_BLK+TPA$L_PARAM,-; Clear the bit corresponding to the
             F353 CF    0CAB 12004                  CLASS_SELECT            ;  desired class.
                   00   0CAE 12005          BBC     #CLS$V_CLUB,-           ; Branch if desired class isn't
         000004E7'EF    0CB0 12006                  TPARSE_BLK+TPA$L_PARAM,-;  CLUSTER
                   07   0CB5 12007                  30$
      00002254'EF   07   D0   0CB6 12008          MOVL    #7,FIRST_DATA_LINE      ; Set first line to be scrolled.
                   50 01 D0   0CBD 12009 30$:     MOVL    #SS$_NORMAL,R0          ; Indicate success.
                   12   11   0CC0 12010          BRB     99$                     ; Branch to common exit.
                        0CC2 12011
000004A9'EF     00000495'EF   DE   0CC2 12012 98$:     MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for class name.
                50 00000000'8F DO   0CCD 12013          MOVL    #SHO$_CRCLS,R0          ; Assume we can't remove the class from
                        0CD4 12014                                                  ;  this report.
```

```
04  0CD4 12015 99$:     RET
    0CD5 12016
    0CD5 12017
```

```
                        ;  .SBTTL          -       ADD_ID
                        ;
                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine adds a system ID to the ID list for systems that will not
                        ;       be included in the display.
                        ;
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       CALLS   #0,ADD_ID
                        ;
                        ; CALLED BY:
                        ;
                        ;       READ_W_PROMPT
                        ;
                        ; INPUTS:
                        ;
                        ;       TKN_DSCR Contains the descriptor for the string which specifies the
                        ;       ID of the system to add to the list.
                        ;
                        ; OUTPUTS:
                        ;
                        ;       Six byte hex ID number is stored in the list of ID numbers for systems
                        ;       to be removed from the display.
                        ;
                        ;       The number of entries in the list is incremented (ID_COUNT).
                        ;
                        ; RETURN STATUS:
                        ;
                        ;       SS$_NORMAL or failed status returned in R0.
                        ;
                        ;--
                        ;
                  0000  ADD_ID::.ENTRY  ^M<>
                        ;
 00000E5C'EF   16       JSB     CONVERT_TO_HEX          ; Convert system ID string to hex value.
       09 50   E9       BLBC    R0,98$                  ; Branch on error.
 00000DA0'EF   16       JSB     ADD_TO_LIST             ; Add the system ID to the list.
       12 50   E8       BLBS    R0,99$                  ; Exit on success.
    50 00000000'8F   D0 98$:    MOVL    #SHOW$_IVREMID,R0       ; "Can't remove system from the display"
000004A9'EF 00000495'EF DE      MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for class name.
                   04   99$:    RET                     ; Return with status.
                        ;
                        ;
```

```
                        0CFC 12067          .SBTTL          -       REM_NODE
                        0CFC 12068
                        0CFC 12069 ;++
                        0CFC 12070 ;
                        0CFC 12071 ; FUNCTIONAL DESCRIPTION:
                        0CFC 12072 ;
                        0CFC 12073 ;       This routine removes a system from the display by converting the node
                        0CFC 12074 ;       name of the system to the system's ID and then adding the system's ID
                        0CFC 12075 ;       to the ID list for systems that will not be included in the display.
                        0CFC 12076 ;
                        0CFC 12077 ;
                        0CFC 12078 ; CALLING SEQUENCE:
                        0CFC 12079 ;
                        0CFC 12080 ;       CALLS   #0,REM_NODE
                        0CFC 12081 ;
                        0CFC 12082 ; CALLED BY:
                        0CFC 12083 ;
                        0CFC 12084 ;       READ_W_PROMPT
                        0CFC 12085 ;
                        0CFC 12086 ; INPUTS:
                        0CFC 12087 ;
                        0CFC 12088 ;       TKN_DSCR Contains the descriptor for the string which specifies the
                        0CFC 12089 ;       node name of the system to remove from the display.
                        0CFC 12090 ;
                        0CFC 12091 ; OUTPUTS:
                        0CFC 12092 ;
                        0CFC 12093 ;       Six byte hex ID number is stored in the list of ID numbers for systems
                        0CFC 12094 ;       to be removed from the display.
                        0CFC 12095 ;
                        0CFC 12096 ;       The number of entries in the list is incremented (ID_COUNT).
                        0CFC 12097 ;
                        0CFC 12098 ; RETURN STATUS:
                        0CFC 12099 ;
                        0CFC 12100 ;       SS$_NORMAL or failed status returned in R0.
                        0CFC 12101 ;
                        0CFC 12102 ;--
                        0CFC 12103
                  0000  0CFC 12104 .ENTRY  REM_NODE,^M<>
                        0CFE 12105
 00000D5B'EF    16      0CFE 12106          JSB     FIND_NODE               ; Find the ID of the system that
                        0D04 12107                                          ;  coresponds to the specified node
                        0D04 12108                                          ;  name.
       09 50    E9      0D04 12109          BLBC    R0,98$                  ; Branch if not found.
 00000DA0'EF    16      0D07 12110 10$:     JSB     ADD_TO_LIST             ; Add the system ID to the list.
       07 50    E8      0D0D 12111          BLBS    R0,99$                  ; Exit on success.
                        0D10 12112
50 00000000'8F  D0      0D10 12113 98$:     MOVL    #SHOW$_IVREMNODE,R0     ; Indicate specified system can not be
                        0D17 12114                                          ;  removed.
                  04    0D17 12115 99$:     RET                             ; Return with status.
                        0D18 12116
                        0D18 12117
```

```
                                                     0D18 12119          .SBTTL          -       REM_ID
                                                     0D18 12120
                                                     0D18 12121 ;++
                                                     0D18 12122 ;
                                                     0D18 12123 ; FUNCTIONAL DESCRIPTION:
                                                     0D18 12124 ;
                                                     0D18 12125 ;       This routine removes adds a system to the display by removing it's
                                                     0D18 12126 ;       system ID from the ID list of systems that will not be included in
                                                     0D18 12127 ;       the display.
                                                     0D18 12128 ;
                                                     0D18 12129 ;
                                                     0D18 12130 ; CALLING SEQUENCE:
                                                     0D18 12131 ;
                                                     0D18 12132 ;       CALLS   #0,REM_ID
                                                     0D18 12133 ;
                                                     0D18 12134 ; CALLED BY:
                                                     0D18 12135 ;
                                                     0D18 12136 ;       READ_W_PROMPT
                                                     0D18 12137 ;
                                                     0D18 12138 ; INPUTS:
                                                     0D18 12139 ;
                                                     0D18 12140 ;       TKN_DSCR Contains the descriptor for the string which specifies the
                                                     0D18 12141 ;       ID of the system to remove from the list.
                                                     0D18 12142 ;
                                                     0D18 12143 ; OUTPUTS:
                                                     0D18 12144 ;
                                                     0D18 12145 ;       The six byte hex ID number stored in the list of ID numbers for systems
                                                     0D18 12146 ;       to be removed from the display is cleared.
                                                     0D18 12147 ;
                                                     0D18 12148 ;       The number of entries in the list is decremented (ID_COUNT).
                                                     0D18 12149 ;
                                                     0D18 12150 ; RETURN STATUS:
                                                     0D18 12151 ;
                                                     0D18 12152 ;       SS$_NORMAL or failed status returned in R0.
                                                     0D18 12153 ;
                                                     0D18 12154 ;--
                                                     0D18 12155
                                          0000       0D18 12156 .ENTRY  REM_ID,^M<>
                                                     0D1A 12157
                  00000E5C'EF    16                  0D1A 12158          JSB     CONVERT_TO_HEX          ; Convert system ID string to hex value.
                        09 50    E9                  0D20 12159          BLBC    R0,98$                  ; Branch on error.
                  00000DFE'EF    16                  0D23 12160          JSB     REMOVE_FROM_LIST        ; Remove the system ID from
                                                     0D29 12161                                          ;  SYS_ID_LIST.
                        12 50    E8                  0D29 12162          BLBS    R0,99$                  ; Exit on success.
                50      00000000'8F    D0           0D2C 12163 98$:     MOVL    #SHOW$_IVADDID,R0       ; "Can't add system to the display"
000004A9'EF     00000495'EF    DE                   0D33 12164          MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for system ID.
                               04                   0D3E 12165 99$:     RET                             ; Return with status.
                                                     0D3F 12166
                                                     0D3F 12167
```

```
                        0D3F 12169          .SBTTL          -       ADD_NODE
                        0D3F 12170
                        0D3F 12171 ;++
                        0D3F 12172 ;
                        0D3F 12173 ; FUNCTIONAL DESCRIPTION:
                        0D3F 12174 ;
                        0D3F 12175 ;       This routine adds a system to the display by converting the node
                        0D3F 12176 ;       name specified into the system ID and removing the system ID
                        0D3F 12177 ;       from the ID list for systems that will not be included in the display.
                        0D3F 12178 ;
                        0D3F 12179 ;
                        0D3F 12180 ; CALLING SEQUENCE:
                        0D3F 12181 ;
                        0D3F 12182 ;       CALLS   #0,ADD_NODE
                        0D3F 12183 ;
                        0D3F 12184 ; CALLED BY:
                        0D3F 12185 ;
                        0D3F 12186 ;       READ_W_ R. PT
                        0D3F 12187 ;
                        0D3F 12188 ; INPUTS:
                        0D3F 12189 ;
                        0D3F 12190 ;       TKN_DSCR Contains the descriptor for the string which specifies the
                        0D3F 12191 ;       node name of the system to remove from the list.
                        0D3F 12192 ;
                        0D3F 12193 ; OUTPUTS:
                        0D3F 12194 ;
                        0D3F 12195 ;       The six byte hex ID number stored in the list of ID numbers for systems
                        0D3F 12196 ;       to be removed from the display is cleared.
                        0D3F 12197 ;
                        0D3F 12198 ;       The number of entries in the list is decremented (ID_COUNT).
                        0D3F 12199 ;
                        0D3F 12200 ; RETURN STATUS:
                        0D3F 12201 ;
                        0D3F 12202 ;       R0 =    SS$_NORMAL
                        0D3F 12203 ;               SHOW$_IVADDNODE if node is unknown or already being displayed.
                        0D3F 12204 ;
                        0D3F 12205 ;--
                        0D3F 12206
                0000    0D3F 12207 .ENTRY  ADD_NODE,^M<>
                        0D41 12208
    00000D5B'EF   16    0D41 12209         JSB     FIND_NODE               ; Find the ID of the system that
                        0D47 12210                                         ;  coresponds to the specified node
                        0D47 12211                                         ;  name.
          09 50   E9    0D47 12212         BLBC    R0,98$                  ; Branch if not found.
    00000DFE'EF   16    0D4A 12213         JSB     REMOVE_FROM_LIST        ; Remove the system ID from the list of
                        0D50 12214                                         ;  systems removed from the display.
          07 50   E8    0D50 12215         BLBS    R0,99$                  ; Exit on success.
50  00000000'8F   D0    0D53 12216 98$:    MOVL    #SHOW$_IVADDNODE,R0     ; Indicate specified node can not be
                        0D5A 12217                                         ;  added.
                04      0D5A 12218 99$:    RET                             ; Return with status.
                        0D5B 12219
                        0D5B 12220
```

```
                     0D5B 12222          .SBTTL          -       FIND_NODE
                     0D5B 12223
                     0D5B 12224 ;++
                     0D5B 12225 ;
                     0D5B 12226 ; FUNCTIONAL DESCRIPTION:
                     0D5B 12227 ;
                     0D5B 12228 ;       This routine is used to convert the node name specified by the
                     0D5B 12229 ;       REMOVE or ADD SYSTEMS/NODE= commands into a system ID.  The
                     0D5B 12230 ;       system blocks in SB buffer are scanned for a node name equal to
                     0D5B 12231 ;       the node name specified.  If a match is found, then the system
                     0D5B 12232 ;       ID of the matching system block is taken from the system block
                     0D5B 12233 ;       and stored in ID_RESULT.  The system ID can then be used to add
                     0D5B 12234 ;       or remove the system ID from SYS_ID_LIST.
                     0D5B 12235 ;
                     0D5B 12236 ; CALLING SEQUENCE:
                     0D5B 12237 ;
                     0D5B 12238 ;       JSB     FIND_NODE
                     0D5B 12239 ;
                     0D5B 12240 ; CALLED BY:
                     0D5B 12241 ;
                     0D5B 12242 ;       REM_NODE
                     0D5B 12243 ;       ADD_NODE
                     0D5B 12244 ;
                     0D5B 12245 ; INPUTS:
                     0D5B 12246 ;
                     0D5B 12247 ;       TKN_DSCR contains a descriptor for the node name string specified.
                     0D5B 12248 ;
                     0D5B 12249 ; OUTPUTS:
                     0D5B 12250 ;
                     0D5B 12251 ;       System ID for the node specified stored in ID_RESULT.
                     0D5B 12252 ;
                     0D5B 12253 ; RETURN STATUS:
                     0D5B 12254 ;
                     0D5B 12255 ;       R0 = SS$_NORMAL if a system block and system ID were found that match
                     0D5B 12256 ;            the node name.
                     0D5B 12257 ;       R0 = 0 if not found.
                     0D5B 12258 ;
                     0D5B 12259 ; REGISTER USAGE:
                     0D5B 12260 ;
                     0D5B 12261 ;       R2  - Used by CMPC.
                     0D5B 12262 ;       R3  - Used by CMPC.
                     0D5B 12263 ;       R10 - Points to the base of a system block in SB buffer.
                     0D5B 12264 ;
                     0D5B 12265 ;--
                     0D5B 12266
                     0D5B 12267 FIND_NODE:
           040C 8F   BB 0D5B 12268          PUSHR   #^M<R2,R3,R10>
      5A 000003ED'EF D0 0D5F 12269          MOVL    VAR_BUF,R10             ; Get the address of the system block
                     0D66 12270                                             ;  buffer.
   00000495'EF   91 0D66 12271 10$:     CMPB    TKN_DSCR,-              ; Is the number of characters specified
         44 AA      0D6C 12272                  SB$T_NODENAME(R10)      ;  the same as the length of the node
                     0D6E 12273                                         ;  name in this system block?
            0F   12 0D6E 12274          BNEQ    20$                     ; If not this can't be the right system
                     0D70 12275                                         ;  block.
0000049A'FF 00000495'EF 29 0D70 12276          CMPC3   TKN_DSCR,@TKN_DSCR+4,-  ; Compare the nodename specified with
         45 AA      0D7B 12277                  SB$T_NODENAME+1(R10)    ;  the nodename in the system block.
            09   13 0D7D 12278          BEQL    30$                     ; Branch if they're the same.
```

```
                     ;*****
                     ;
                     ; Get the address of the next system block.
                     ;
                     ;*****

                     20$:    MOVL    (R10),R10               ; Get the address of the next system
                                                             ;  block.
                             BNEQ    10$                     ; Loop of there's more SB's.
                             CLRL    R0                      ; Otherwise, indicate failure,
                             BRB     99$                     ;  and return with error status.
                     30$:    MOVL    SB$B_SYSTEMID(R10),-    ; Store the system ID for the matching
000022DC'EF                  ID_RESULT                       ;  system block in ID_RESULT.
                             MOVW    SB$B_SYSTEMID+4(R10),-  ;
000022E0'EF                  ID_RESULT+4                     ;
                             MOVL    #SS$_NORMAL,R0          ; Indicate success.
                     99$:    POPR    #^M<R2,R3,R10>
                             RSB
```

```
                     ODA0 12294          .SBTTL          -       ADD_TO_LIST
                     ODA0 12295
                     ODA0 12296 ;++
                     ODA0 12297 ;
                     ODA0 12298 ; FUNCTIONAL DESCRIPTION:
                     ODA0 12299 ;
                     ODA0 12300 ;       The routine disables a system from being displayed by adding it's
                     ODA0 12301 ;       system ID to the list of systems not to be displayed.  It searches
                     ODA0 12302 ;       the list at SYS_ID_LIST to make sure that the system id is not
                     ODA0 12303 ;       already in the list.  If it isn't, then it inserts the system ID
                     ODA0 12304 ;       in the first empty entry.
                     ODA0 12305 ;
                     ODA0 12306 ; CALLING SEQUENCE:
                     ODA0 12307 ;
                     ODA0 12308 ;       JSB     ADD_TO_LIST
                     ODA0 12309 ;
                     ODA0 12310 ; CALLED BY:
                     ODA0 12311 ;
                     ODA0 12312 ;       ADD_ID
                     ODA0 12313 ;       REM_NODE
                     ODA0 12314 ;
                     ODA0 12315 ; INPUTS:
                     ODA0 12316 ;
                     ODA0 12317 ;       SYS_ID_LIST contains a list of system ID's for systems removed from
                     ODA0 12318 ;       the display.
                     ODA0 12319 ;       ID_RESULT contains the ID of the system to remove add to the list.
                     ODA0 12320 ;       ID_COUNT contains the number of system entered in the list.
                     ODA0 12321 ;
                     ODA0 12322 ; OUTPUTS:
                     ODA0 12323 ;
                     ODA0 12324 ;       System ID for entered in SYS_ID_LIST.
                     ODA0 12325 ;       ID_COUNT updated.
                     ODA0 12326 ;
                     ODA0 12327 ; RETURN STATUS:
                     ODA0 12328 ;
                     ODA0 12329 ;       R0 = SS$_NORMAL or failed status if the entry is already in the
                     ODA0 12330 ;           list or if there is no more room in the list.
                     ODA0 12331 ;
                     ODA0 12332 ; REGISTER USAGE:
                     ODA0 12333 ;
                     ODA0 12334 ;       R2  - Used as an index for comparing the six bytes of the system ID.
                     ODA0 12335 ;       R4  - Used as a pointer into the SYS_ID_LIST.
                     ODA0 12336 ;       R5  - Used to count down the total number of possible entries.
                     ODA0 12337 ;       R6  - Used to save the address of the first empty entry found.
                     ODA0 12338 ;
                     ODA0 12339 ;--
                     ODA0 12340
                     ODA0 12341 ADD_TO_LIST:
          0074 8F    BB  ODA0 12342          PUSHR   #^M<R2,R4,R5,R6>
               50    D4  ODA4 12343          CLRL    R0                      ; Assume failure.
    54 00002278'EF   DE  ODA6 12344          MOVAL   SYS_ID_LIST,R4          ; Get the address of the list.
               56    D4  ODAD 12345          CLRL    R6                      ; Assume no empty entry.
          55    10    D0  ODAF 12346          MOVL    #ID_LIST_ENTRYS,R5      ; Get the number of entries to search.
          52    02    D0  ODB2 12347          MOVL    #2,R2                   ; R2 = word index.
    64 000022DC'EF   D1  ODB5 12348 10$:     CMPL    ID_RESULT,(R4)          ; Compare the specified result with
                         ODBC 12349                                          ;  the entry in the list.
                0A   12  ODBC 12350          BNEQ    20$                     ; Branch if no match.
```

```
                6442  000022E0'EF   B1  0DBE 12351          CMPW    ID_RESULT+4,(R4)[R2]    ; Compare all 6 bytes.
                      31   13  0DC6 12352          BEQL    99$                     ; Branch if the system ID is already in
                                0DC8 12353                                          ;  the list.
                      56   D5  0DC8 12354 20$:     TSTL    R6                      ; Have we saved the address of the
                                0DCA 12355                                          ;  empty entry in R6?
                      0C   12  0DCA 12356          BNEQ    30$                     ; Branch if so.
                      64   D5  0DCC 12357          TSTL    (R4)                    ; Is this entry empty?
                      08   12  0DCE 12358          BNEQ    30$                     ; Branch if not.
                    6442   B5  0DD0 12359          TSTW    (R4)[R2]                ; Check all 6 bytes.
                      03   12  0DD3 12360          BNEQ    30$                     ; Branch if entry isn't empty.
                56    54   D0  0DD5 12361          MOVL    R4,R6                   ; Save the address of the empty entry.
                54    06   C0  0DD8 12362 30$:     ADDL    #6,R4                   ; Point to next entry.
                  D7  55   F5  0DDB 12363          SOBGTR  R5,10$                  ; Loop for entire list.
                      56   D5  0DDE 12364          TSTL    R6                      ; Did we find an empty entry.
                      17   13  0DE0 12365          BEQL    99$                     ; Branch if not.
              86      000022DC'EF   D0  0DE2 12366          MOVL    ID_RESULT,(R6)+         ; Otherwise, save the system ID in the
              66      000022E0'EF   B0  0DE9 12367          MOVW    ID_RESULT+4,(R6)        ;  list.
                      000022D8'EF   D6  0DF0 12368          INCL    ID_COUNT                ; Increase count of entries in the
                                0DF6 12369                                          ;  list.
                50    01   D0  0DF6 12370          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                0074  8F   BA  0DF9 12371 99$:     POPR    #^M<R2,R4,R5,R6>
                      05   0DFD 12372          RSB
                                0DFE 12373
                                0DFE 12374
```

```
                     ODFE 12376          .SBTTL          -       REMOVE_FROM_LIST
                     ODFE 12377
                     ODFE 12378 ;++
                     ODFE 12379 ;
                     ODFE 12380 ; FUNCTIONAL DESCRIPTION:
                     ODFE 12381 ;
                     ODFE 12382 ;       This routine clears the entry for a system in the ID list of systems
                     ODFE 12383 ;       that are not to be included in the display thereby adding the system
                     ODFE 12384 ;       to the display.
                     ODFE 12385 ;
                     ODFE 12386 ; CALLING SEQUENCE:
                     ODFE 12387 ;
                     ODFE 12388 ;       JSB     REMOVE_FROM_LIST
                     ODFE 12389 ;
                     ODFE 12390 ; CALLED BY:
                     ODFE 12391 ;
                     ODFE 12392 ;       REM_ID
                     ODFE 12393 ;       ADD_NODE
                     ODFE 12394 ;
                     ODFE 12395 ; INPUTS:
                     ODFE 12396 ;
                     ODFE 12397 ;       ID_RESULT contains the system ID to match.
                     ODFE 12398 ;       SYS_ID_LIST contains a list of system ID's for systems removed from
                     ODFE 12399 ;       the display.
                     ODFE 12400 ;       ID_COUNT contains the number of entries in the list.
                     ODFE 12401 ;
                     ODFE 12402 ; OUTPUTS:
                     ODFE 12403 ;
                     ODFE 12404 ;       Entry in SYS_ID_LIST coresponding to the the system ID specified
                     ODFE 12405 ;       is cleared.
                     ODFE 12406 ;       The number of entries in the list stored at ID_COUNT is decremented.
                     ODFE 12407 ;
                     ODFE 12408 ; RETURN STATUS:
                     ODFE 12409 ;
                     ODFE 12410 ;       R0 = SS$_NORMAL or failed status if entry wasn't found.
                     ODFE 12411 ;
                     ODFE 12412 ; REGISTER USAGE:
                     ODFE 12413 ;
                     ODFE 12414 ;       R4  Contains the address of the entry as returned from the
                     ODFE 12415 ;       FIND_SYS_ID routine.
                     ODFE 12416 ;
                     ODFE 12417 ;--
                     ODFE 12418
                     ODFE 12419 REMOVE_FROM_LIST:
           10   BB   ODFE 12420          PUSHR   #^M<R4>
00000E16'EF   16   0E00 12421          JSB     FIND_SYS_ID             ; Is the entry in the list?
         0A 50   E9   0E06 12422          BLBC    R0,99$                  ; Branch if not.
           84   D4   0E09 12423          CLRL    (R4)+                   ; Zero out the entry.
           64   B4   0E0B 12424          CLRW    (R4)                    ; All six bytes.
000022D8'EF   D7   0E0D 12425          DECL    ID_COUNT                ; Decrease number of entries in the
                     0E13 12426                                          ;  list.
           10   BA   0E13 12427 99$:    POPR    #^M<R4>
                05   0E15 12428          RSB
                     0E16 12429
                     0E16 12430
```

```
                        0E16 12432          .SBTTL          -       FIND_SYS_ID
                        0E16 12433
                        0E16 12434 ;++
                        0E16 12435 ;
                        0E16 12436 ; FUNCTIONAL DESCRIPTION:
                        0E16 12437 ;
                        0E16 12438 ;       This routine locates a system ID in the ID list.
                        0E16 12439 ;
                        0E16 12440 ;
                        0E16 12441 ; CALLING SEQUENCE:
                        0E16 12442 ;
                        0E16 12443 ;       JSB     FIND_SYS_ID
                        0E16 12444 ;
                        0E16 12445 ; CALLED BY:
                        0E16 12446 ;
                        0E16 12447 ;       REM_ID
                        0E16 12448 ;       CHECK_ID
                        0E16 12449 ;
                        0E16 12450 ; INPUTS:
                        0E16 12451 ;
                        0E16 12452 ;       ID_RESULT contains the system ID to match.
                        0E16 12453 ;       SYS_ID_LIST contains a list of system ID's.
                        0E16 12454 ;       ID_COUNT contains the number of entries in the list.
                        0E16 12455 ;
                        0E16 12456 ; OUTPUTS:
                        0E16 12457 ;
                        0E16 12458 ;       Address of entry returned in R4.
                        0E16 12459 ;
                        0E16 12460 ; RETURN STATUS:
                        0E16 12461 ;
                        0E16 12462 ;       R0 = SS$_NORMAL if the system ID is in the list.
                        0E16 12463 ;
                        0E16 12464 ; REGISTER USAGE:
                        0E16 12465 ;
                        0E16 12466 ;--
                        0E16 12467
                        0E16 12468 FIND_SYS_ID:
                  0C    BB  0E16 12469          PUSHR   #^M<R2,R3>
                  50    D4  0E18 12470          CLRL    R0                      ; Assume failure.
        54 00002278'EF  DE  0E1A 12471          MOVAL   SYS_ID_LIST,R4          ; Get the address of the list.
        53 000022D8'EF  D0  0E21 12472          MOVL    ID_COUNT,R3             ; Get the number of entries in the list.
                  2F    13  0E28 12473          BEQL    99$                     ; Branch if list is empty.
              52    02  D0  0E2A 12474          MOVL    #2,R2                   ; Index = word.
        64 000022DC'EF  D1  0E2D 12475 10$:     CMPL    ID_RESULT,(R4)          ; Does the entry match the designated
                        0E34 12476                                              ;  system?
                  0F    12  0E34 12477          BNEQ    20$                     ; Branh if not.
      6442 000022E0'EF  B1  0E36 12478          CMPW    ID_RESULT+4,(R4)[R2]    ; Check all six bytes.
                  05    12  0E3E 12479          BNEQ    20$                     ; Branch if no match.
              50    01  D0  0E40 12480          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                  14    11  0E43 12481          BRB     99$                     ; Done.
                  64    D5  0E45 12482 20$:     TSTL    (R4)                    ; Is the entry empty?
                  0A    12  0E47 12483          BNEQ    30$                     ; Branch if not.
                6442    B5  0E49 12484          TSTW    (R4)[R2]                ; Check all six bytes.
                  05    12  0E4C 12485          BNEQ    30$                     ; Branch if not empty.
              54    06  C0  0E4E 12486          ADDL    #6,R4                   ; Point to next entry.
                  DA    11  0E51 12487          BRB     10$                     ; Loop if empty entry.
              54    06  C0  0E53 12488 30$:     ADDL    #6,R4                   ; Point to next entry.
```

```
D4 53   F5  0E56 12489          SOBGTR  R3,10$                  ; Loop for all valid entries in the
                0E59 12490                                      ;  list.
   0C   BA  0E59 12491 99$:     POPR    #^M<R2,R3>
        05  0E5B 12492          RSB
            0E5C 12493
            0E5C 12494
```

```
                     0E5C 12496          .SBTTL          -       CONVERT_TO_HEX
                     0E5C 12497
                     0E5C 12498 ;++
                     0E5C 12499 ;
                     0E5C 12500 ; FUNCTIONAL DESCRIPTION:
                     0E5C 12501 ;
                     0E5C 12502 ;       This routine converts the system ID string at TKN_DSCR to a six
                     0E5C 12503 ;       byte hex value and stores the value at ID_RESULT.
                     0E5C 12504 ;
                     0E5C 12505 ;
                     0E5C 12506 ; CALLING SEQUENCE:
                     0E5C 12507 ;
                     0E5C 12508 ;       JSB     CONVERT_TO_HEX
                     0E5C 12509 ;
                     0E5C 12510 ; CALLED BY:
                     0E5C 12511 ;
                     0E5C 12512 ;       REM_ID
                     0E5C 12513 ;       ADD_ID
                     0E5C 12514 ;
                     0E5C 12515 ; INPUTS:
                     0E5C 12516 ;
                     0E5C 12517 ;       TKN_DSCR contains a descriptor for the system ID string.
                     0E5C 12518 ;
                     0E5C 12519 ; OUTPUTS:
                     0E5C 12520 ;
                     0E5C 12521 ;       ID_RESULT contains the system ID.
                     0E5C 12522 ;
                     0E5C 12523 ; RETURN STATUS:
                     0E5C 12524 ;
                     0E5C 12525 ;       SS$_NORMAL or failed status returned in R0.
                     0E5C 12526 ;
                     0E5C 12527 ; REGISTER USAGE:
                     0E5C 12528 ;
                     0E5C 12529 ;--
                     0E5C 12530
                     0E5C 12531          .ENABL  LSB
                     0E5C 12532
                     0E5C 12533 CONVERT_TO_HEX:
          07FC 8F    BB  0E5C 12534          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
               50    D4  0E60 12535          CLRL    R0                      ; Assume failure.
52   00000129'EF    DE  0E62 12536          MOVAL   SB$B_SYSTEMID_DSCR,R2   ; Get the address of the field
                         0E69 12537                                          ;  descriptor for the system ID.
     02    05 A2    91  0E69 12538          CMPB    FD$B_FORMAT(R2),#HEX    ; Is the system ID being displayed in
                         0E6D 12539                                          ;  hexidecimal?
               24    12  0E6D 12540          BNEQU   DECIMAL_ID              ; Branch if not.
0C   00000495'EF    B1  0E6F 12541          CMPW    TKN_DSCR,#12            ; Are there more than 12 digits in the
                         0E76 12542                                          ;  system ID string that was specified?
                         0E76 12543          JGTRU   99$                     ; Error if so.
                         0E7B 12544 ;
                         0E7B 12545 ; Convert the string into a 6 byte hex number.
                         0E7B 12546 ;
               06    DD  0E7B 12547          PUSHL   #6                      ; Six byte result.
     000022DC'EF    DF  0E7D 12548          PUSHAL  ID_RESULT               ; Address to receive result.
     00000495'EF    DF  0E83 12549          PUSHAL  TKN_DSCR                ; String descriptor.
00000000'GF    03    FB  0E89 12550          CALLS   #3,G^OTS$CVT_TZ_L       ; Convert the string.
              006B    31  0E90 12551          BRW     99$                     ; Jump to common code.
                         0E93 12552
```

```
                                0E93 12553 DECIMAL_ID:
                                0E93 12554
                                0E93 12555 ;
                                0E93 12556 ; Convert the decimal string to a six byte hex number.
                                0E93 12557 ;
                                0E93 12558
 52   00000495'EF    3C        0E93 12559          MOVZWL  TKN_DSCR,R2             ; Get the number of digits in the
                                0E9A 12560                                          ;  string.
 55   00000499'EF    D0        0E9A 12561          MOVL    TKN_DSCR+4,R5           ; Get the address of the string.
          14   52    D1        0EA1 12562          CMPL    R2,#20                  ; Are there more that twenty digits in
                                0EA4 12563                                          ;  the string?
                                0EA4 12564          JGTRU   99$                     ; Error if so.
      53  52   03    9C        0EA9 12565          ROTL    #3,R2,R3                ; Multiply number of digits by 8 to get
                                0EAD 12566                                          ;  quadword index.
 54   0000056A'EF    DE        0EAD 12567          MOVAL   DEC_TAB_END,R4          ; Get the address of the end of the
                                0EB4 12568                                          ;  decimal conversion table.
          54   53    C2        0EB4 12569          SUBL    R3,R4                   ; Get the address of the first quadword
                                0EB7 12570                                          ;  in the table to use for conversion.
      000022DC'EF    D4        0EB7 12571          CLRL    ID_RESULT               ; Clear the result of conversion.
      000022E0'EF    D4        0EBD 12572          CLRL    ID_RESULT+4             ;
          56   84    7D        0EC3 12573 10$:     MOVQ    (R4)+,R6                ; Get the quadword.
      53  65   30    83        0EC6 12574          SUBB3   #^X30,(R5),R3           ; Convert the ascii digit to decimal.
               2A    13        0ECA 12575          BEQLU   40$                     ; Ignore if digit is zero.
                                0ECC 12576          JLSSU   99$                     ; Branch if not valid decimal number.
          09   53    91        0ED1 12577          CMPB    R3,#9                   ; Check high limit.
               28    1A        0ED4 12578          BGTRU   99$                     ; Branch if not valid decimal number.
                                0ED6 12579 ;
                                0ED6 12580 ; Multiply the quadword in R6,R7 by the byte in R3 and add the result to
                                0ED6 12581 ; ID_RESULT.
                                0ED6 12582 ;
          53   53    9A        0ED6 12583          MOVZBL  R3,R3                   ; Convert byte to longword.
 58   00   56   53    7A        0ED9 12584          EMUL    R3,R6,#0,R8             ; Multiply low longword.
               03    18        0EDE 12585          BGEQ    20$                     ; Branch if result is positive.
          59   53    C0        0EE0 12586          ADDL    R3,R9                   ; Otherwise, compensate for unsighned
                                0EE3 12587                                          ;  bias.
 59   59   57   53    7A        0EE3 12588 20$:     EMUL    R3,R7,R9,R9             ; Multiply high longword.
  000022DC'EF    58    C0        0EE8 12589 30$:     ADDL    R8,ID_RESULT            ; Add the resulting quadword to
  000022E0'EF    59    D8        0EEF 12590          ADWC    R9,ID_RESULT+4          ;  ID_RESULT.
               55    D6        0EF6 12591 40$:     INCL    R5                      ; Point to next byte of string.
            C8 52    F5        0EF8 12592          SOBGTR  R2,10$                  ; Loop for all digits in the string.
          50   01    D0        0EFB 12593          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                                0EFE 12594
          07FC 8F    BA        0EFE 12595 99$:     POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
               05    05        0F02 12596          RSB
                                0F03 12597
                                0F03 12598          .DSABL  LSB
                                0F03 12599
                                0F03 12600
```

```
                     0F03 12602          .SBTTL          -       SET_WIDTH
                     0F03 12603
                     0F03 12604 ;++
                     0F03 12605 ;
                     0F03 12606 ; FUNCTIONAL DESCRIPTION:
                     0F03 12607 ;
                     0F03 12608 ;       This routine is used to change the number of columns used by a
                     0F03 12609 ;       particular field.  It changes the value in the field descriptor at
                     0F03 12610 ;       location FD$B_FIELD_SIZE after checking that the desired width does not
                     0F03 12611 ;       exceed the minimum or maximum values for the field.
                     0F03 12612 ;
                     0F03 12613 ; CALLING SEQUENCE:
                     0F03 12614 ;
                     0F03 12615 ;       CALLS   #0,SET_WIDTH
                     0F03 12616 ;
                     0F03 12617 ; CALLED BY:
                     0F03 12618 ;
                     0F03 12619 ;       READ_W_PROMPT
                     0F03 12620 ;
                     0F03 12621 ; INPUTS:
                     0F03 12622 ;
                     0F03 12623 ;       Desired width stored at location VALUE.
                     0F03 12624 ;       Address of the field descriptor for the selected field stored in
                     0F03 12625 ;       KEY_DSCR.
                     0F03 12626 ;
                     0F03 12627 ; OUTPUTS:
                     0F03 12628 ;
                     0F03 12629 ;       Value at location VALUE stored at offset FD$B_FIELD_SIZE in the field
                     0F03 12630 ;       descriptor for the selected field.
                     0F03 12631 ;
                     0F03 12632 ; RETURN STATUS:
                     0F03 12633 ;
                     0F03 12634 ;       SS$_NORMAL or failed status returned in R0.
                     0F03 12635 ;
                     0F03 12636 ; REGISTER USAGE:
                     0F03 12637 ;
                     0F03 12638 ;       R4 -    Points to the base address of the field descriptor for the
                     0F03 12639 ;               selected field.
                     0F03 12640 ;--
                     0F03 12641
                0010 0F03 12642 .ENTRY  SET_WIDTH,^M<R4>
54   0000054B'EF   D0 0F05 12643        MOVL    KEY_DSCR,R4             ; Get the field descriptor address.
     0000054F'EF   91 0F0C 12644        CMPB    VALUE,-                 ; Is the desired size smaller than the
           01 A4      0F12 12645                FD$B_MIN_FLD_SIZE(R4)   ;  minimum allowed for this field?
              13   1F 0F14 12646        BLSSU   10$                     ; Branch if so.
     0000054F'EF   91 0F16 12647        CMPB    VALUE,-                 ; Is the desired size larger than the
           03 A4      0F1C 12648                FD$B_MAX_FLD_SIZE(R4)   ;  maximum allowed for this field?
              0F   1A 0F1E 12649        BGTRU   20$                     ; Branch if so.
     0000054F'EF   90 0F20 12650        MOVB    VALUE,-                 ; Store the new field size.
              64        0F26 12651                FD$B_FIELD_SIZE(R4)     ;
              0A   11 0F27 12652        BRB     99$                     ; Jump to common code.
           01 A4   90 0F29 12653 10$:   MOVB    FD$B_MIN_FLD_SIZE(R4),- ; Store the minimum field size.
              64        0F2C 12654                FD$B_FIELD_SIZE(R4)     ;
              04   11 0F2D 12655        BRB     99$                     ; Jump to common code.
           03 A4   90 0F2F 12656 20$:   MOVB    FD$B_MAX_FLD_SIZE(R4),- ; Store the maximum field size.
              64        0F32 12657                FD$B_FIELD_SIZE(R4)     ;
                   04   0F33 12658 99$:   RET
```

```
0F34 12659
0F34 12660
```

```
                        I  8
0F34 12662          .SBTTL          -       SET_FORMAT
0F34 12663 ;++
0F34 12664 ;
0F34 12665 ; FUNCTIONAL DESCRIPTION:
0F34 12666 ;
0F34 12667 ;       This routine is used to change the output format of a particular field
0F34 12668 ;       to decimal or hexadecimal.  It changes the value in the field
0F34 12669 ;       descriptor at location FD$B_FORMAT after checking that the current
0F34 12670 ;       format is not ascii or ascic.  FD$B_MAX_DATA_SIZE is set to the maximum
0F34 12671 ;       number of columns needed to display the data in the new format.  This
0F34 12672 ;       value is also compared with FD$B_TITLE_SIZE (the number of characters
0F34 12673 ;       in the field title) and the larger of the two is saved as the maximum
0F34 12674 ;       number of columns necessary to display the field (FD$B_MAX_FLD_SIZE).
0F34 12675 ;
0F34 12676 ;       Note:   This routine will return an error if the number of bytes of
0F34 12677 ;               data needed to display the field exceeds eight and the format
0F34 12678 ;               is being changed from hexadecimal to decimal.  This is because
0F34 12679 ;               the format routines will not format a value larger than a
0F34 12680 ;               quadword in decimal.
0F34 12681 ;
0F34 12682 ; CALLING SEQUENCE:
0F34 12683 ;
0F34 12684 ;       CALLS   #0,SET_FORMAT
0F34 12685 ;
0F34 12686 ; CALLED BY:
0F34 12687 ;
0F34 12688 ;       READ_W_PROMPT
0F34 12689 ;
0F34 12690 ; INPUTS:
0F34 12691 ;
0F34 12692 ;       Desired format type stored at location TPARSE_BLK+TPA$L_PARAM.
0F34 12693 ;       Address of the field descriptor for the selected field stored in
0F34 12694 ;       KEY_DSCR.
0F34 12695 ;
0F34 12696 ; OUTPUTS:
0F34 12697 ;
0F34 12698 ;       Value at location TPARSE_BLK+TPA$L_PARAM stored at offset
0F34 12699 ;       FD$B_FORMAT in the field descriptor for the selected field.
0F34 12700 ;
0F34 12701 ;       Value at offset FD$B_MAX_DATA_SIZE in the field descriptor set to the
0F34 12702 ;       maximum number of columns needed to display the data.
0F34 12703 ;
0F34 12704 ;       Value at offset FD$B_MAX_FLD_SIZE in the field descriptor set to the
0F34 12705 ;       maximum number of columns needed to display the field.
0F34 12706 ;
0F34 12707 ; RETURN STATUS:
0F34 12708 ;
0F34 12709 ;       SS$_NORMAL or failed status returned in R0.
0F34 12710 ;
0F34 12711 ; REGISTER USAGE:
0F34 12712 ;
0F34 12713 ;       R3 -    Used to hold the number of byte of SCS data needed to display
0F34 12714 ;               the field.
0F34 12715 ;       R4 -    Points to the base address of the field descriptor for the
0F34 12716 ;               selected field.
0F34 12717 ;       R5 -    Pointer to data length to number of columns conversion table.
0F34 12718 ;--
```

```
                        0F34 12719
                0038    0F34 12720 .ENTRY  SET_FORMAT,^M<R3,R4,R5>
54  0000054B'EF   D0    0F36 12721         MOVL    KEY_DSCR,R4                 ; Get the field descriptor address.
    000004E7'EF   91    0F3D 12722         CMPB    TPARSE_BLK+TPA$L_PARAM,-    ; Is the format already set to the
          05 A4         0F43 12723                 FD$B_FORMAT(R4)             ;  desired type?
             50   13    0F45 12724         BEQL    99$                         ; Return if so.
       01    05 A4   91  0F47 12725         CMPB    FD$B_FORMAT(R4),#ASCII      ; Is the field formated in ascii?
             48   13    0F4B 12726         BEQL    98$                         ; Error if so.
       04    05 A4   91  0F4D 12727         CMPB    FD$B_FORMAT(R4),#ASCIC      ; Is the field formated in ascii using
                         0F51 12728                                             ;  a counted ascii string?
             42   13    0F51 12729         BEQL    98$                         ; Error if so.
       53    04 A4   9A  0F53 12730         MOVZBL  FD$B_LENGTH(R4),R3          ; Get the number of bytes of SCS data
                         0F57 12731                                             ;  needed to display this field.
    000004E7'EF   91    0F57 12732         CMPB    TPARSE_BLK+TPA$L_PARAM,-    ; Is the new format decimal?
                03       0F5D 12733                 #DEC                        ;
                13   12  0F5E 12734         BNEQ    10$                         ; Branch if not.
          08    53   91  0F60 12735         CMPB    R3,#8                       ; Is the SCS data more that 8 bytes?
                30   14  0F63 12736         BGTR    98$                         ; Error if so.
55  000004A9'EF   DE    0F65 12737         MOVAL   CVT_TABLE,R5                ; Point to conversion table.
    02 A4    6543   90   0F6C 12738         MOVB    (R5)[R3],-                  ; Save the maximum number of columns
                         0F71 12739                 FD$B_MAX_DATA_SIZE(R4)      ;  needed to display the field in
                         0F71 12740                                             ;  decimal.
                05   11  0F71 12741         BRB     20$                         ; Jump to common code.
 02 A4    53    53   81  0F73 12742 10$:    ADDB3   R3,R3,-                     ; Save the maximum number of columns
                         0F78 12743                 FD$B_MAX_DATA_SIZE(R4)      ;  needed to display the field in
                         0F78 12744                                             ;  hexadecimal.
    000004E7'EF   90    0F78 12745 20$:    MOVB    TPARSE_BLK+TPA$L_PARAM,-    ; Set the new format type.
          05 A4         0F7E 12746                 FD$B_FORMAT(R4)             ;
          02 A4   91    0F80 12747         CMPB    FD$B_MAX_DATA_SIZE(R4),-    ; Are more columns required to display
          18 A4         0F83 12748                 FD$B_TITLE_SIZE(R4)         ;  the data than to display the field
                         0F85 12749                                             ;  title?
             07   1F    0F85 12750         BLSSU   30$                         ; Branch if not.
    02 A4    90   0F87 12751         MOVB    FD$B_MAX_DATA_SIZE(R4),-    ; Make the maximum field size the same
    03 A4         0F8A 12752                 FD$B_MAX_FLD_SIZE(R4)       ;  as the maximum data size.
             09   11  0F8C 12753         BRB     99$                         ; Return with success.
    18 A4    90   0F8E 12754 30$:    MOVB    FD$B_TITLE_SIZE(R4),-       ; Make the maximum field size the same
    03 A4         0F91 12755                 FD$B_MAX_FLD_SIZE(R4)       ;  as the title size.
             02   11  0F93 12756         BRB     99$                         ; Return with success.
             50   D4  0F95 12757 98$:    CLRL    R0                          ; Indicate failure.
                  04  0F97 12758 99$:    RET
                         0F98 12759
                         0F98 12760
```

```
                                0F98 12762          .SBTTL          -       SET_INTERVAL
                                0F98 12763 ;++
                                0F98 12764 ;
                                0F98 12765 ; FUNCTIONAL DESCRIPTION:
                                0F98 12766 ;
                                0F98 12767 ;       This routine converts the number of seconds string entered in the SET
                                0F98 12768 ;       INTERVAL command to binary time and stores it at location DELAY.
                                0F98 12769 ;
                                0F98 12770 ; CALLING SEQUENCE:
                                0F98 12771 ;
                                0F98 12772 ;       CALLS   #0,SET_INTERVAL
                                0F98 12773 ;
                                0F98 12774 ; CALLED BY:
                                0F98 12775 ;
                                0F98 12776 ;       READ_W_PROMPT
                                0F98 12777 ;
                                0F98 12778 ; INPUTS:
                                0F98 12779 ;
                                0F98 12780 ;       Desired number of seconds stored at location VALUE.
                                0F98 12781 ;
                                0F98 12782 ; OUTPUTS:
                                0F98 12783 ;
                                0F98 12784 ;       Binary representation of desired interval stored at location DELAY.
                                0F98 12785 ;
                                0F98 12786 ; RETURN STATUS:
                                0F98 12787 ;
                                0F98 12788 ;       SS$_NORMAL returned in R0.
                                0F98 12789 ;
                                0F98 12790 ;--
                                0F98 12791
                         0000   0F98 12792 .ENTRY  SET_INTERVAL,^M<>
                                0F9A 12793
    0000054F'EF    0000054F'EF    3C   0F9A 12794          MOVZWL  VALUE,VALUE             ; Make sure VALUE is a longword.
00  0000054F'EF    FF676980 8F    7A   0FA5 12795          EMUL    #-10000000,VALUE,#0,-   ; Convert the number of seconds to
                   000021AC'EF          0FB1 12796                  DELAY                   ;  binary time.
                        50    01    D0   0FB6 12797          MOVL    #SS$_NORMAL,R0          ; Return success.
                                    04   0FB9 12798          RET                             ; Return with status.
                                0FBA 12799
                                0FBA 12800
```

```
                                0FBA 12802          .SBTTL          -       SET_SCREEN
                                0FBA 12803 ;++
                                0FBA 12804 ;
                                0FBA 12805 ; FUNCTIONAL DESCRIPTION:
                                0FBA 12806 ;
                                0FBA 12807 ;       This routine sets the width of the terminal screen to 80 or 132
                                0FBA 12808 ;       columns.  If the desired width is 80 or less than the screen is set to
                                0FBA 12809 ;       80 columns.  If the desired width is 81 or greater than the screen is
                                0FBA 12810 ;       set to 132 columns.
                                0FBA 12811 ;       If this routine is not being called while initialization or exit is
                                0FBA 12812 ;       in progress and is being run on a video terminal than the display
                                0FBA 12813 ;       header is redrawn.
                                0FBA 12814 ;
                                0FBA 12815 ; CALLING SEQUENCE:
                                0FBA 12816 ;
                                0FBA 12817 ;       CALLS   #0,SET_SCREEN
                                0FBA 12818 ;
                                0FBA 12819 ; CALLED BY:
                                0FBA 12820 ;
                                0FBA 12821 ;       INIT
                                0FBA 12822 ;       MAIN_ERR
                                0FBA 12823 ;       READ_W_PROMPT
                                0FBA 12824 ;
                                0FBA 12825 ; INPUTS:
                                0FBA 12826 ;
                                0FBA 12827 ;       Desired screen width stored at location VALUE.
                                0FBA 12828 ;
                                0FBA 12829 ; OUTPUTS:
                                0FBA 12830 ;
                                0FBA 12831 ;       SCREEN_WIDTH is set to 80 or 132.
                                0FBA 12832 ;
                                0FBA 12833 ; IMPLICIT OUTPUTS:
                                0FBA 12834 ;
                                0FBA 12835 ;       Terminal is set to 80 or 132 column mode.
                                0FBA 12836 ;
                                0FBA 12837 ; RETURN STATUS:
                                0FBA 12838 ;
                                0FBA 12839 ;       SS$_NORMAL or failed status returned in R0.
                                0FBA 12840 ;
                                0FBA 12841 ;--
                                0FBA 12842
                         0000   0FBA 12843 .ENTRY  SET_SCREEN,^M<>
                                0FBC 12844
                   50    01    D0  0FBC 12845          MOVL    #SS$_NORMAL,R0          ; Assume success.
0050 8F    0000054F'EF    B1    0FBF 12846          CMPW    VALUE,#NARROW           ; Is the desired screen width > 80 ?
                      44    1A  0FC8 12847          BGTRU   10$                     ; Branch if so.
00002222'EF    0050 8F    B1    0FCA 12848          CMPW    #NARROW,SCREEN_WIDTH    ; Is the terminal width already 80?
                                0FD3 12849          JEQL    99$                     ; Return if it is.
00002222'EF    0050 8F    B0    0FD8 12850          MOVW    #NARROW,SCREEN_WIDTH    ; Set number of columns to 80.
  76 00002247'EF    01    E1    0FE1 12851          BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if terminal isn't a DECcrt.
          0000014E'EF    DF    0FE9 12852          PUSHAL  DCRT_NARROW_DSCR        ; Push address of descriptor to set
                                0FEF 12853                                          ;  terminal into 80 column mode.
  5E 00002247'EF    04    E0    0FEF 12854          BBS     #DVT$V_AVO,OUT_DEV,20$  ; Branch if terminal is a DECcrt with
                                0FF7 12855                                          ;  advanced video option.
     00002260'EF    18    D0    0FF7 12856          MOVL    #24,LAST_LINE           ; Save the number of the last line
                                0FFE 12857                                          ;  (bottom command line) on the
                                0FFE 12858                                          ;  terminal.
```

```
                0000225C'EF   17   D0  0FFE 12859          MOVL    #23,CMD_LINE                ; Save the number of the next-to-last
                                  1005 12860                                                   ;  line (top command line) on the
                                  1005 12861                                                   ;  terminal.
                00002258'EF   16   D0  1005 12862          MOVL    #22,LAST_DATA_LINE          ; Save the number of the last line on
                                  100C 12863                                                   ;  the terminal to be used to display
                                  100C 12864                                                   ;  data.
                              47   11  100C 12865          BRB     20$                         ; Jump to common code.
                  60 00002247'EF   00   E0  100E 12866 10$:  BBS     #DVT$V_VT52,OUT_DEV,99$     ; Branch if terminal is VT52 type.
              00002222'EF   0084   8F   B1  1016 12867          CMPW    #WIDE,SCREEN_WIDTH          ; Is the terminal width already 132?
                              55   13  101F 12868          BEQL    99$                         ; Return if it is.
              00002222'EF   0084   8F   B0  1021 12869          MOVW    #WIDE,SCREEN_WIDTH          ; Set number of columns to 132.
                44 00002247'EF   01   E1  102A 12870          BBC     #DVT$V_DCRT,OUT_DEV,99$     ; Branch if terminal isn't a DECcrt.
                15 00002247'EF   04   E0  1032 12871          BBS     #DVT$V_AVO,OUT_DEV,15$      ; Branch if DECcrt with advanced video
                                  103A 12872                                                   ;  option.
                00002260'EF   0E   D0  103A 12873          MOVL    #14,LAST_LINE               ; Save the number of the last line
                                  1041 12874                                                   ;  (bottom command line) on the
                                  1041 12875                                                   ;  terminal.
                0000225C'EF   0D   D0  1041 12876          MOVL    #13,CMD_LINE                ; Save the number of the next-to-last
                                  1048 12877                                                   ;  line (top command line) on the
                                  1048 12878                                                   ;  terminal.
                00002258'EF   0C   D0  1048 12879          MOVL    #12,LAST_DATA_LINE          ; Save the number of the last line on
                                  104F 12880                                                   ;  the terminal to be used to display
                                  104F 12881                                                   ;  data.
                     0000015B'EF   DF  104F 12882 15$:  PUSHAL  DCRT_WIDE_DSCR              ; Push address of descriptor to set
                                  1055 12883                                                   ;  terminal into 132 column mode.
                00000000'GF   01   FB  1055 12884 20$:  CALLS   #1,G^LIB$PUT_OUTPUT         ; Set the terminal.
                          17   50   E9  105C 12885          BLBC    R0,99$                      ; Branch on error.
                              01   E0  105F 12886 30$:  BBS     #MF$V_EXIT_FLAG,-           ; Branch if exit flag is set.
                  0F 00000476'EF                  1061 12887          MISC_FLAGS,99$              ;
                              00   E0  1067 12888          BBS     #MF$V_INIT_FLAG,-           ; Or if INIT flag is set.
                  07 00000476'EF                  1069 12889          MISC_FLAGS,99$              ;
                00002348'EF   00   FB  106F 12890          CALLS   #0,REFRESH                  ; Refresh the screen.
                              04   1076 12891 99$:  RET
                                  1077 12892
                                  1077 12893
```

```
                     1077 12895          .SBTTL          -       GET_HELP
                     1077 12896 ;++
                     1077 12897 ;
                     1077 12898 ; FUNCTIONAL DESCRIPTION:
                     1077 12899 ;
                     1077 12900 ;       This routine is called when the user asks for help.  The remainder
                     1077 12901 ;       of the string is passed to the LBR$OUTPUT_HELP routine.
                     1077 12902 ;
                     1077 12903 ; CALLING SEQUENCE:
                     1077 12904 ;
                     1077 12905 ;       CALLS   #0,GET_HELP
                     1077 12906 ;
                     1077 12907 ; CALLED BY:
                     1077 12908 ;
                     1077 12909 ;       READ_W_PROMPT
                     1077 12910 ;
                     1077 12911 ; IMPLICIT_INPUTS:
                     1077 12912 ;
                     1077 12913 ;       TPARSE_BLK+TPA$L_STRINGPTR points to the first keyword in the command
                     1077 12914 ;       buffer following the "HELP" command string.
                     1077 12915 ;
                     1077 12916 ;       CMD_LINE_DSCR contains the number of bytes in the entire command line.
                     1077 12917 ;
                     1077 12918 ; IMPLICIT OUTPUTS:
                     1077 12919 ;
                     1077 12920 ;       Help txt is output to the terminal.
                     1077 12921 ;
                     1077 12922 ; RETURN STATUS:
                     1077 12923 ;
                     1077 12924 ;       SS$_NORMAL or failed status returned in R0.
                     1077 12925 ;
                     1077 12926 ;--
                     1077 12927
                0800 1077 12928 .ENTRY  GET_HELP,^M<R11>
                     1079 12929
                     1079 12930 ;
                     1079 12931 ; Store the number of bytes remaining in the command string in HELP_DSCR.
                     1079 12932 ;
    000020D7'EF   C3 1079 12933         SUBL3   CMD_LINE_DSCR+4,-           ; Get the number of bytes we've
    000004D3'EF      107F 12934                 TPARSE_BLK+TPA$L_STRINGPTR,-;  already parsed.
              50     1084 12935                 R0
                     1085 12936
 000020D3'EF  50   C3 1085 12937         SUBL3   R0,CMD_LINE_DSCR,-          ; Save the number of bytes remaining.
    00000553'EF      108C 12938                 HELP_DSCR                   ;
                     1091 12939 ;
                     1091 12940 ; Store the address of remainder of the command string in HELP_DSCR+4.
                     1091 12941 ;
    000004DB'EF   D0 1091 12942         MOVL    TPARSE_BLK+TPA$L_TOKENPTR,-
    00000557'EF      1097 12943                 HELP_DSCR+4
                     109C 12944
 00002248'EF  06   D0 109C 12945         MOVL    #HEADER_END-1,LINE_NUM      ; Set the scrolling region to the area
            104B'  30 10A3 12946         BSBW    DRAW_LAST_LINE              ; Draw bottom line of the display box
                     10A6 12947                                             ;  as the bottom line of the display
                     10A6 12948                                             ;  header.
                     10A6 12949         JLBC    R0,99$                      ; Branch on error.
09 00002247'EF   01  E1 10AC 12950         BBC     #DVT$V_DCRT,OUT_DEV,10$ ; Branch if terminal is not a DECcrt.
    00001A03'EF   16 10B4 12951         JSB     SET_REGION_HELP             ; Establish a scrolling region.
```

```
                 6B 50    E9  10BA 12952                 BLBC    R0,99$                  ; Branch on error.
00002248'EF      07       D0  10BD 12953 10$:          MOVL    #HEADER_END,LINE_NUM    ; Set cursor to line after header.
     0000224C'EF 3F       10C4 12954                   PUSHAW  COL_NUM                 ; Erase all the data.
     00002248'EF 3F       10CA 12955                   PUSHAW  LINE_NUM                ;
00000000'GF      02       FB  10D0 12956               CALLS   #2,G^LIB$ERASE_PAGE     ;
                 4E 50    E9  10D7 12957               BLBC    R0,99$                  ; Branch on error.
     00000000'GF 9F       10DA 12958                   PUSHAL  G^LIB$GET_INPUT         ; Push address of input routine.
     0000047D'EF DF       10E0 12959                   PUSHAL  HELP_FLAGS              ; Push address of flags.
     0000010C'EF DF       10E6 12960                   PUSHAL  LIBRARY_NAME            ; Push address of descriptor for the
                          10EC 12961                                                   ;  library name.
     00000553'EF DF       10EC 12962                   PUSHAL  HELP_DSCR               ; Push address of descriptor for the
                          10F2 12963                                                   ;  help keywords.
     00002222'EF 3C       10F2 12964                   MOVZWL  SCREEN_WIDTH,-          ; Save width of screen.
     0000222C'EF          10F8 12965                   OUTPUT_WIDTH                    ;
     0000222C'EF DF       10FD 12966                   PUSHAL  OUTPUT_WIDTH            ; Push address of longword containing
                          1103 12967                                                   ;  the output width.
     00000000'GF DF       1103 12968                   PUSHAL  G^LIB$PUT_OUTPUT        ; Push address of output routine.
00000000'GF      06       FB  1109 12969               CALLS   #6,G^LBR$OUTPUT_HELP    ; Call librarian help routine.
                 15 50    E9  1110 12970               BLBC    R0,99$                  ; Branch on error.
03 00002247'EF   01       E1  1113 12971               BBC     #DVT$V_DCRT,OUT_DEV,20$ ; Branch if terminal is not a DECcrt.
                    08B0'  30  111B 12972               BSBW    SET_REGION_FULL         ; Set scrolling region to use full
                           111E 12973                                                   ;  screen.
00002348'EF      00       FB  111E 12974 20$:          CALLS   #0,REFRESH              ; Refresh the screen.
                 50   01  D0  1125 12975               MOVL    #SS$_NORMAL,R0
                          04  1128 12976 99$:          RET
                              1129 12977
                              1129 12978
```

```
                     1129 12980          .SBTTL          -       RESET
                     1129 12981 ;++
                     1129 12982 ;
                     1129 12983 ; FUNCTIONAL DESCRIPTION:
                     1129 12984 ;
                     1129 12985 ;       This routine restores the default classes and fields for the cluster
                     1129 12986 ;       and local reports.  It also reselects the cluster report, reenables
                     1129 12987 ;       any systems that have been removed from the report, and resets the
                     1129 12988 ;       screen to 80 columns if run on a DECcrt.
                     1129 12989 ;
                     1129 12990 ; CALLING SEQUENCE:
                     1129 12991 ;
                     1129 12992 ;       CALLS   #0,RESET
                     1129 12993 ;
                     1129 12994 ; CALLED BY:
                     1129 12995 ;
                     1129 12996 ;       READ_W_PROMPT
                     1129 12997 ;
                     1129 12998 ; OUTPUTS
                     1129 12999 ;
                     1129 13000 ;       Cluster bitmap restored using copy saved by the INIT routine.
                     1129 13001 ;       Local bitmap restored using copy saved by the INIT routine.
                     1129 13002 ;       CURRENT_REPORT set to cluster.
                     1129 13003 ;       Default cluster report classes stored in CLASS_SELECT.
                     1129 13004 ;       Default local report classes stored in LOCAL_SAVE.
                     1129 13005 ;
                     1129 13006 ; RETURN STATUS:
                     1129 13007 ;
                     1129 13008 ;       None.
                     1129 13009 ;
                     1129 13010 ;--
                     1129 13011
                00FC 1129 13012 .ENTRY  RESET,^M<R2,R3,R4,R5,R6,R7>
                     112B 13013
0000054F'EF  00000050 8F   D0 112B 13014          MOVL    #80,VALUE               ; Screen width = 80 columns.
                     1136 13015
                     1136 13016 ;
                     1136 13017 ; Set default classes for each report.
                     1136 13018 ;
            00370006 8F   D0 1136 13019          MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
            00002204'EF      113C 13020                  CLUST_SAVE
    EEB7 CF 00000180 8F   D0 1141 13021          MOVL    #CLS$M_LPRT!CLS$M_ERR,CLASS_SELECT
                     114A 13022 ;
                     114A 13023 ; Restore saved copy of cluster bitmap.
                     114A 13024 ;
                55   02   D0 114A 13025          MOVL    #CLUSTER_MAP_LEN,R5     ; Get the length of the cluster bitmap.
          56 00000000'EF   DE 114D 13026          MOVAL   CLUSTER_BITMAP,R6       ; Get the address of the cluster
                     1154 13027                                                  ;  bitmap.
          57 0000000C'EF   DE 1154 13028          MOVAL   CL_SAVE_BITMAP,R7       ; Get the address of the saved copy of
                     115B 13029                                                  ;  the cluster bitmap.
              86   87   D0 115B 13030 60$:     MOVL    (R7)+,(R6)+             ; Restore the cluster bitmap.
                FA 55   F4 115E 13031          SOBGEQ  R5,60$                  ;
                     1161 13032
                     1161 13033 ;
                     1161 13034 ; Restore the field sizes for the cluster report.
                     1161 13035 ;
                     1161 13036
```

```
                57  00000000'EF   DE  1161 13037          MOVAL   CLUSTER_SIZE_ARRAY,R7   ; Get the address of the cluster field
                                      1168 13038                                          ;  size array.
                56  00000000'EF   DE  1168 13039          MOVAL   CLUSTER_TABLE,R6        ; Get the address of the cluster
                                      116F 13040                                          ;  address table.
                55  00000046 8F   D0  116F 13041          MOVL    #NUM_CLUSTER_DSCRS,R5   ; Get the number of field descriptors
                                      1176 13042                                          ;  for the cluster report.
                          53      D4  1176 13043          CLRL    R3                      ; Clear index.
                    52    6643    D0  1178 13044 70$:     MOVL    (R6)[R3],R2             ; Get the address of the field
                                      117C 13045                                          ;  descriptor.
                    62    6743    90  117C 13046          MOVB    (R7)[R3],-              ; Copy the original field size back
                                      1180 13047                    FD$B_FIELD_SIZE(R2)   ;  into the field descriptor.
                F4  53    55      F2  1180 13048          AOBLSS  R5,R3,70$               ; Loop for each field descriptor in the
                                      1184 13049                                          ;  cluster report.
                                      1184 13050 ;
                                      1184 13051 ; Restore saved copy of the local ports bitmap.
                                      1184 13052 ;
                    55    00      D0  1184 13053          MOVL    #LOCAL_MAP_LEN,R5       ; Get the length of the local bitmap.
                56  00000000'EF   DE  1187 13054          MOVAL   LOCAL_BITMAP,R6         ; Get the address of the local bitmap.
                57  00000004'EF   DE  118E 13055          MOVAL   LOC_SAVE_BITMAP,R7      ; Get the address of the saved copy of
                                      1195 13056                                          ;  the local bitmap.
                    86    87      D0  1195 13057 80$:     MOVL    (R7)+,(R6)+             ; Restore the local ports bitmap.
                      FA  55      F4  1198 13058          SOBGEQ  R5,80$                  ;
                                      119B 13059
                                      119B 13060 ;
                                      119B 13061 ; Restore the field sizes for the local report.
                                      119B 13062 ;
                                      119B 13063
                57  00000000'EF   DE  119B 13064          MOVAL   LOCAL_SIZE_ARRAY,R7     ; Get the address of the local field
                                      11A2 13065                                          ;  size array.
                56  00000000'EF   DE  11A2 13066          MOVAL   LOCAL_TABLE,R6          ; Get the address of the local
                                      11A9 13067                                          ;  address table.
                    55    19      D0  11A9 13068          MOVL    #NUM_LOCAL_DSCRS,R5     ; Get the number of field descriptors
                                      11AC 13069                                          ;  for the local report.
                          53      D4  11AC 13070          CLRL    R3                      ; Clear index.
                    52    6643    D0  11AE 13071 90$:     MOVL    (R6)[R3],R2             ; Get the address of the field
                                      11B2 13072                                          ;  descriptor.
                    62    6743    90  11B2 13073          MOVB    (R7)[R3],-              ; Copy the original field size back
                                      11B6 13074                    FD$B_FIELD_SIZE(R2)   ;  into the field descriptor.
                F4  53    55      F2  11B6 13075          AOBLSS  R5,R3,90$               ; Loop for each field descriptor in the
                                      11BA 13076                                          ;  local report.
                                      11BA 13077
00  00002278'EF       00      2C  11BA 13078          MOVC5   #0,SYS_ID_LIST,#0,-     ; Restore any systems that have been
00002278'EF       0060  8F        11C2 13079                    #ID_LIST_LEN,SYS_ID_LIST; removed from the display.
                000022D8'EF       D4  11CA 13080          CLRL    ID_COUNT                ; Clear the number of system removed
                                      11D0 13081                                          ;  from the display.
                                      11D0 13082
                0000047A'EF       94  11D0 13083          CLRB    SCROLL_FLAGS            ; Clear the scroll flags.
                0000047C'EF       94  11D6 13084          CLRB    CLSTR_SCROLL_FLAGS      ; Clear the flags that will be retored
                                      11DC 13085                                          ;  as the scroll flags after call to
                                      11DC 13086                                          ;  SHOW_CLUSTER.
                00000464'EF       D4  11DC 13087          CLRL    CLSTR_H_SCROLL          ; Clear all the scroll counters.
                00000468'EF       D4  11E2 13088          CLRL    LOCAL_H_SCROLL          ;
                0000045C'EF       D4  11E8 13089          CLRL    V_SCROLL                ;
                          01      8A  11EE 13090          BICB    #RPT$M_CLSTR,-          ; Force the SHOW_CLUSTER routine to
                      EE0D  CF        11F0 13091                  CURRENT_REPORT          ;  reset the report by setting report
                                      11F3 13092                                          ;  to the local report.
                52  00000476'EF   DO  11F3 13093          MOVL    MISC_FLAGS,R2           ; Save copy of miscellaneous flags.
```

```
                    01   C8  11FA 13094          BISL    #MFSM_INIT_FLAG,-       ; Set init flag so we don't draw the
 00000476'EF            11FC 13095                  MISC_FLAGS                      ;  local report.
FDB4 CF    00   FB  1201 13096          CALLS   #0,SET_SCREEN           ; Set the screen width.
                    1206 13097
                    1206 13098 ASSUME   MFSV_INIT_FLAG EQ 0
                    1206 13099
01   00    52   F0  1206 13100          INSV    R2,#MFSV_INIT_FLAG,#1,- ; Restore init flag to original state.
 00000476'EF        120A 13101                  MISC_FLAGS                      ;
F715 CF    00   FB  120F 13102          CALLS   #0,SHOW_CLUSTER         ; Set default report to clusters.
                04  1214 13103          RET
                    1215 13104
                    1215 13105
```

```
                                1215 13107          .SBTTL          -       EXECUTE
                                1215 13108 ;++
                                1215 13109 ;
                                1215 13110 ; FUNCTIONAL DESCRIPTION:
                                1215 13111 ;
                                1215 13112 ;       This routine is used to execute a command procedure when requested by
                                1215 13113 ;       the user (@filespec), to execute a startup initialization file
                                1215 13114 ;       when Show Cluster if first envoked, and to read commands when the input
                                1215 13115 ;       is from a file.
                                1215 13116 ;
                                1215 13117 ; CALLING SEQUENCE:
                                1215 13118 ;
                                1215 13119 ;       CALLS   #0,EXECUTE
                                1215 13120 ;
                                1215 13121 ; CALLED BY:
                                1215 13122 ;
                                1215 13123 ;       INIT                                    ; Execute initialization file.
                                1215 13124 ;       READ_W_PROMPT                           ; Execute command procedure.
                                1215 13125 ;       END_REPORT                              ; Get command input from a file.
                                1215 13126 ;
                                1215 13127 ; RETURN STATUS:
                                1215 13128 ;
                                1215 13129 ;       SS$_NORMAL or falied status returned in R0.
                                1215 13130 ;
                                1215 13131 ;--
                                1215 13132
                        003C    1215 13133 .ENTRY  EXECUTE,^M<R2,R3,R4,R5>
                                1217 13134
    20  000000CC'EF   00    2C  1217 13135          MOVC5   #0,IN_FILE_SPEC,#^A/ /,-; Pad the filespec with blanks.
                0050 8F         121F 13136                  #IN_FILE_SPEC_LEN,-     ;
                000000CC'EF     1222 13137                  IN_FILE_SPEC            ;
                                1227 13138
                          00    E1  1227 13139 3$:      BBC     #MF$V_INIT_FLAG,-       ; Branch if initialization not in
             12 00000476'EF         1229 13140                  MISC_FLAGS,4$           ;  progress.
0000012E'FF     0000012A'EF   28    122F 13141          MOVC3   INIT_NAME,@INIT_NAME+4,-; Store logical name in the file spec.
                000000CC'EF         123A 13142                  IN_FILE_SPEC            ;
                          2A    11  123F 13143          BRB     7$                      ; Jump to common.
                                1241 13144
          12 00002246'EF   05    E1  1241 13145 4$:      BBC     #DVT$V_DISK,IN_DEV,5$   ; Branch if input isn't from a file.
                000000DE'EF   28    1249 13146          MOVC3   IN_DEV_NAME,-           ; Otherwise SYS$INPUT is the filespec.
                000000E6'EF         124F 13147                  IN_DEV_NAME+8,-         ;
                000000CC'EF         1254 13148                  IN_FILE_SPEC            ;
                          10    11  1259 13149          BRB     7$                      ; Jump to common.
                                125B 13150
                                125B 13151 ;
                                125B 13152 ; Get the filename from the command line and store in in the filespec.
                                125B 13153 ;
                                125B 13154
                000004CF'EF   28    125B 13155 5$:      MOVC3   TPARSE_BLK+TPA$L_STRINGCNT,-
                000004D3'FF         1261 13156                  @TPARSE_BLK+TPA$L_STRINGPTR,-
                000000CC'EF         1266 13157                  IN_FILE_SPEC
                                126B 13158
                                126B 13159 7$:      $PARSE  FAB = INFAB             ; Expand the filespec.
                                1278 13160          JLBC    R0,20$                  ; Branch on error.
                                127E 13161
                                127E 13162          $OPEN   FAB = INFAB             ; Open the file.
                                128B 13163          JLBC    R0,20$                  ; Branch on error.
```

```
00000476'EF   00000040 8F   C8  1291 13164          BISL    #MF$M_IN_OPEN,MISC_FLAGS; Indicate file is open.
                            1297 13165
                            1297 13166          $CONNECT -                      ; Connect to input.
                            1297 13167                  RAB = INRAB             ;
                      60 50 E9  12A9 13168          BLBC    R0,20$                  ; Branch on error.
                            12AC 13169
                            12AC 13170 10$:     $GET    RAB = INRAB             ; Get a record.
                      50 50 E9  12B9 13171          BLBC    R0,20$                  ; Branch on error.
                            12BC 13172
00000310'EF   000002EE'EF BO  12BC 13173          MOVW    INRAB+RAB$W_RSZ,REC_DSCR; Get the record size.
              00000310'EF DF  12C7 13174          PUSHAL  REC_DSCR                ; Convert record to uppercase.
              00000310'EF DF  12CD 13175          PUSHAL  REC_DSCR                ;
        00000000'GF   02  FB  12D3 13176          CALLS   #2,G^STR$UPCASE         ;
                            12DA 13177
                            12DA 13178 ;
                            12DA 13179 ; Parse the record.
                            12DA 13180 ;
                            12DA 13181
000004CF'EF   000002EE'EF BO  12DA 13182          MOVW    INRAB+RAB$W_RSZ,TPARSE_BLK+TPA$L_STRINGCNT
000004D3'EF   000002F4'EF DO  12E5 13183          MOVL    INRAB+RAB$L_RBF,TPARSE_BLK+TPA$L_STRINGPTR
              00000000'EF DF  12F0 13184          PUSHAL  CMD_KEY
              00000000'EF DF  12F6 13185          PUSHAL  CMD_STATE
              000004C7'EF DF  12FC 13186          PUSHAL  TPARSE_BLK
        00000000'GF   03  FB  1302 13187          CALLS   #3,G^LIB$TPARSE
                      A0 50 E8  1309 13188          BLBS    R0,10$                  ; Loop for entire file.
                            130C 13189
                         06 E1  130C 13190 20$:     BBC     #MF$V_IN_OPEN,-         ; Branch if the file isn't open.
                2A 00000476'EF  130E 13191                  MISC_FLAGS,25$          ;
                      52   50 DO  1314 13192          MOVL    R0,R2                   ; Save error status.
                            1317 13193          $CLOSE  FAB = INFAB             ; Close the file.
                      3A 50 E9  1324 13194          BLBC    R0,99$                  ; Branch on error.
00000476'EF   00000040 8F   CA  1327 13195          BICL    #MF$M_IN_OPEN,MISC_FLAGS; Indicate file is closed.
                      50   52 DO  1332 13196          MOVL    R2,R0                   ; Restore previous status.
              00000000'8F   50 D1  1335 13197          CMPL    R0,#RMS$_EOF            ; Was error condition END-OF-FILE?
                         11 13  133C 13198          BEQL    30$                     ; Branch if so.
              00000000'8F   50 D1  133E 13199 25$:     CMPL    R0,#RMS$_DEV            ; Was error bad or inappropriate device?
                         1A 12  1345 13200          BNEQ    99$                     ; Branch if not.
                         00 E1  1347 13201          BBC     #MF$V_INIT_FLAG,-       ; Ok if called by INIT, assume SHC$INIT
                12 00000476'EF  1349 13202                  MISC_FLAGS,99$          ;  not defined.
                      50   01 DO  134F 13203 30$:     MOVL    #SS$_NORMAL,R0          ; Indicate success.
                         00 E0  1352 13204          BBS     #MF$V_INIT_FLAG,-       ; Branch if initialization pass.
                07 00000476'EF  1354 13205                  MISC_FLAGS,99$          ;
              00002348'EF   00  FB  135A 13206          CALLS   #0,REFRESH              ; Update the screen.
                            04  1361 13207 99$:     RET
                            1362 13208
                            1362 13209
                            1362 13210          .END    START
```

$$$AST	= 00000000			ADD_SYS_QUAL	0000017F	R	11
$$$CNT	= 00000003			ADD_SYS_TYPE	000001CC	R	11
$$$CURSIZ	= 000001C4			ADD_SYS_TYPE_LOOP	000001DA	R	11
$$$FLG	= FFFFFFFF			ADD_SYS_TYPE_STRING	000001E6	R	11
$$$KEY	= 0000004B			ADD_TO_CLST	00000DA0	R	10
$$$KFG	= FFFFFFFF			ADRS_TABLE	00002264	R	03
$$$MOD	= 00000000			ASCIC	= 00000004		
$$$NEWSIZ	= 000001D0			ASCIC_DISPLAY	000026AF	R	0F
$$$TMP	= 000001E8	R	13	ASCII	= 00000001		
$$.TAB	= 000002CC	R	03	ASCII_DISPLAY	000025C9	R	0F
$$.TABEND	= 00000310	R	03	ASC_TIM_DSCR	00001F8E	R	03
$$.TMP	= 00000000			AST_DEL	00001A1E	RG	0F
$$.TMP1	= 00000001			A_TIME	000021B4	R	03
$$.TMP2	= 000000CF			BDT_WAITS_ADRS	00000859	R	09
$$.TMPX	= 00000004	R	04	BEGIN_QUAL	00000067	R	02
$$.TMPX1	= 0000000C			BITMAP_ADRS	00002268	R	03
$$KEYTAB	= 00000000	R	12	BLANK_LINES	0000043C	R	03
$$T1	= 00000001			BLKS_REQ_ADRS	000007B5	R	09
ACKR_SQ_ADRS	000002CA	R	09	BLKS_SENT_ADRS	00000793	R	09
ACK_CM_ADRS	00000329	R	09	BOTTOM_T	= 00000076		
ADD	00000048	R	11	BOT_CMD_LINE	00000471	R	03
ADD_ALL_FIELDS	00000BCA	RG	10	BRDCST	0000045B	R	0F
ADD_CIR	00000260	R	11	BUFADR	00000000	R	03
ADD_CIR_QUAL	00000271	R	11	BUFFER_SIZE	00002210	R	03
ADD_CIR_TYPE	0000028E	R	11	BUFF_DESCRIPTORS_ADRS	0000003B	R	0E
ADD_CIR_TYPE_LOOP	0000029C	R	11	BUFF_MSG	000001DF	R	02
ADD_CIR_TYPE_STRING	000002A8	R	11	BUFSIZE	00000004	R	03
ADD_CLUSTER	000000BA	R	11	BUF_COUNT	000003E4	R	03
ADD_CLUSTER_ALL	000000CB	R	11	BUF_QUO	000004B5	R	03
ADD_CON	000002D3	R	11	BUILD_DECCRT_BOX	00001F9D	R	0F
ADD_CON_QUAL	000002E4	R	11	BUILD_VT52_BOX	00001F44	R	0F
ADD_CON_TYPE	00000301	R	11	CABLE_STATUS_ADRS	00000585	R	09
ADD_CON_TYPE_LOOP	0000030F	R	11	CB_STAT_STRG	0000028C	R	02
ADD_CON_TYPE_STRING	0000031B	R	11	CB_STAT_STRNG_LEN	= 00000007		
ADD_COUNTERS	000000F6	R	11	CDT$C_LISTEN	= 00000001		
ADD_COUNTERS_QUAL	00000107	R	11	CDT$C_OPEN	= 00000002		
ADD_CREDITS	00000114	R	11	CDT$L_BYTMAPD	= 00000094		
ADD_CREDITS_QUAL	00000125	R	11	CDT$L_BYTMAPD_DSCR	000007FE	R	09
ADD_ERRORS	00000132	R	11	CDT$L_BYTREQD	= 00000090		
ADD_ERRORS_QUAL	00000143	R	11	CDT$L_BYTREQD_DSCR	000007DE	R	09
ADD_FIELD	00000A15	RG	10	CDT$L_BYTSENT	= 00000088		
ADD_ID	00000CD5	RG	10	CDT$L_BYTSENT_DSCR	000007BE	R	09
ADD_ID_LOOP	000001B2	R	11	CDT$L_CDTLST	= 0000006C		
ADD_LOCAL_PORTS	00000150	R	11	CDT$L_DGDISCARD	= 00000078		
ADD_LOCAL_PORTS_QUAL	00000161	R	11	CDT$L_DGDISCARD_DSCR	00000715	R	09
ADD_MEMBERS	000000D8	R	11	CDT$L_DGRCVD	= 00000074		
ADD_MEMBERS_QUAL	000000E9	R	11	CDT$L_DGRCVD_DSCR	000006F4	R	09
ADD_NEXT_CLASS	0000009E	R	11	CDT$L_DGSENT	= 00000070		
ADD_NODE	00000D3F	RG	10	CDT$L_DGSENT_DSCR	000006D3	R	09
ADD_NODE_LOOP	00000246	R	11	CDT$L_LCONID	= 00000018		
ADD_ONE_SPACE	00002443	R	0F	CDT$L_LCONID_DSCR	0000066D	R	09
ADD_SPACES	00002449	R	0F	CDT$L_MSGINPOT	= 00000000		
ADD_SYS	0000016E	R	11	CDT$L_MSGRCVD	= 00000080		
ADD_SYS_ID	000001A4	R	11	CDT$L_MSGRCVD_DSCR	00000759	R	09
ADD_SYS_ID_STRING	000001BE	R	11	CDT$L_MSGSENT	= 0000007C		
ADD_SYS_NODE	00000238	R	11	CDT$L_MSGSENT_DSCR	00000737	R	09
ADD_SYS_NODE_STRING	00000252	R	11	CDT$L_RCONID	= 00000014		

```
CDT$L_RCONID_DSCR           0000068F R    09
CDT$L_REQDATS             = 0000008C
CDT$L_REQDATS_DSCR          0000079D R    09
CDT$L_SCSMSG              = 0000002C
CDT$L_SNDDATS             = 00000084
CDT$L_SNDDATS_DSCR          0000077B R    09
CDT$W_BLKSTATE            = 0000002A
CDT$W_BLKSTATE_DSCR         000006B1 R    09
CDT$W_INITLREC            = 00000048
CDT$W_INITLREC_DSCR         000008E2 R    09
CDT$W_MINREC              = 00000044
CDT$W_MINREC_DSCR           00000903 R    09
CDT$W_MINSEND             = 0000004A
CDT$W_MINSEND_DSCR          00000880 R    09
CDT$W_PENDREC             = 00000046
CDT$W_PENDREC_DSCR          000008C1 R    09
CDT$W_QBDT_CNT            = 0000009A
CDT$W_QBDT_CNT_DSCR         00000841 R    09
CDT$W_QCR_CNT             = 00000098
CDT$W_QCR_CNT_DSCR          00000820 R    09
CDT$W_REC                 = 00000042
CDT$W_REC_DSCR              000008A1 R    09
CDT$W_SEND                = 00000040
CDT$W_SEND_DSCR             00000863 R    09
CDT$W_STATE               = 00000028
CDT$W_STATE_DSCR            0000064B R    09
CDT_BLKSTATE_STRNG          000003C7 R    02
CDT_BUF_PNTR                00000023 R    10
CDT_COUNT                   000003EB R    03
CDT_DSCR                    0000003F R    10
CDT_FIRST_LOOP              00002FE5 R    0F
CDT_IN                    = 00000005
CDT_LOOP                    00002FAE R    0F
CDT_PFX                     0000049B R    02
CDT_SNAP                    00002F5A R    0F
CDT_SNAP_DONE               00003009 R    0F
CDT_SNAP_END                00003018 R    0F
CDT_STATE_STRNG             0000035A R    02
CHAR_BUF_LEN              = 0000000C
CHECK_ASCII                 000026E6 R    0F
CHECK_DIF                   00002D3D R    0F
CHECK_ID                    000014F8 R    0F
CHECK_MORE                  000018D4 R    0F
CHECK_NUM_CDT               0000147A R    0F
CHECK_NUM_PB                0000144C R    0F
CHECK_POS                   000014A8 R    0F
CHECK_QUE                   000032F4 R    0F
CHECK_VALID                 000026CF R    0F
CIBD$L_LINK               = 0000000C
CIBDT$L_FREEBD            = FFFFFFF4
CIR_STATE_ADRS              00000541 R    09
CLSM_BRDCST               = 000000C1
CLSM_CMD                  = 00000002
CLSM_ERROR                = 00000004
CLSM_SCROLL               = 00000008
CLSV_BRDCST               = 00000000
CLSV_CMD                  = 00000001
CLSV_ERROR                = 00000002
CLSV_SCROLL               = 00000003
CLASS                     = 00000100
CLASS_BOX_BUF               0000076B R    03
CLASS_BOX_DSCR              00000763 R    03
CLASS_BOX_LEN             = 00000200
CLASS_BUF                   0000159B R    03
CLASS_BUF_LEN             = 00000400
CLASS_DSCR                  00001593 R    03
CLASS_NAMES                 0000023B R    02
CLASS_SELECT                00000001 R    10
CLEAR_SCREEN                000023D0 R    0F
CLI$DCL_PARSE               ******** X    0F
CLI$GET_VALUE               ******** X    0F
CLI$PRESENT                 ******** X    0F
CLS$M_750_SYS             = 00010000
CLS$M_780_SYS             = 00020000
CLS$M_785_SYS             = 00040000
CLS$M_ALL_CIR             = 00000608
CLS$M_ALL_CON             = 00001810
CLS$M_ALL_SYS             = 00370002
CLS$M_ANY_CLUS            = 0000007F
CLS$M_ANY_SCS             = 0000007E
CLS$M_CIR                 = 00000008
CLS$M_CLUB                = 00000001
CLS$M_CNT                 = 00000020
CLS$M_CON                 = 00000010
CLS$M_CRD                 = 00000040
CLS$M_CSB                 = 00000004
CLS$M_ERR                 = 00000100
CLS$M_HSC50_SYS           = 00100000
CLS$M_HSC60_SYS           = 00200000
CLS$M_LPRT                = 00000080
CLS$M_NOOPEN_CIR          = 00000400
CLS$M_NOOPEN_CON          = 00001000
CLS$M_OPEN_CIR            = 00000200
CLS$M_OPEN_CON            = 00000800
CLS$M_SYS                 = 00000002
CLS$V_750_SYS             = 00000010
CLS$V_780_SYS             = 00000011
CLS$V_785_SYS             = 00000012
CLS$V_790_SYS             = 00000013
CLS$V_CIR                 = 00000003
CLS$V_CLUB                = 00000000
CLS$V_CNT                 = 00000005
CLS$V_CON                 = 00000004
CLS$V_CRD                 = 00000006
CLS$V_CSB                 = 00000002
CLS$V_ERR                 = 00000008
CLS$V_HSC50_SYS           = 00000014
CLS$V_HSC60_SYS           = 00000015
CLS$V_LPRT                = 00000007
CLS$V_NOOPEN_CIR          = 0000000A
CLS$V_NOOPEN_CON          = 0000000C
CLS$V_OPEN_CIR            = 00000009
CLS$V_OPEN_CON            = 0000000B
CLS$V_SYS                 = 00000001
```

```
CLSTR_H_SCROLL                  00000464 R      03
CLSTR_SCROLL_FLAGS              0000047C R      03
CLUSGL_CLUB                     ********   X    0F
CLUBSC_LENGTH                 = 000001A8
CLUBSL_FLAGS                  = 0000001C
CLUBSL_FLAGS_DSCR               00000063 R      09
CLUBSQ_FTIME                  = 0000002C
CLUBSQ_FTIME_DSCR               000000C5 R      09
CLUBSQ_LST_TIME               = 0000003C
CLUBSQ_LST_TIME_DSCR            000000E4 R      09
CLUBST_QDNAME                 = 000000B8
CLUBST_QDNAME_DSCR              00000043 R      09
CLUBSV_QF_VOTE                = 00000019
CLUBSW_MEMSEQ                 = 000000AC
CLUBSW_MEMSEQ_DSCR              000000A6 R      09
CLUBSW_QDVOTES                = 000000AE
CLUBSW_QDVOTES_DSCR             00000083 R      09
CLUBSW_QUORUM                 = 00000020
CLUBSW_QUORUM_DSCR              00000000 R      09
CLUBSW_VOTES                  = 00000022
CLUBSW_VOTES_DSCR               00000022 R      09
CLUB_BOF                        0000004B R      10
CLUB_HEADER_SIZE                000021C8 R      03
CLUB_IN                       = 00000001
CLUB_NUM_FIELDS                 00000427 R      03
CLUSTER_BITMAP                  00000000 R      06
CLUSTER_KEYTAB                  00000000 R      08
CLUSTER_MAP_LEN               = 00000002
CLUSTER_MSG                     0000018A R      02
CLUSTER_SIZE_ARRAY              00000000 R      05
CLUSTER_SNAP                    00002E2E R      0F
CLUSTER_TABLE                   00000000 R      07
CLUSTER_TAB_LEN               = 00000118
CLUST_SAVE                      00002204 R      03
CL_QDVOTES_ADRS                 0000009B R      09
CL_QUORUM_ADRS                  00000018 R      09
CL_SAVE_BITMAP                  0000000C R      06
CL_VOTES_ADRS                   0000003A R      09
CMDS_QUEUED_ADRS                00000231 R      0E
CMD_BUF                         000020DB R      03
CMD_BUF_LEN                   = 00000078
CMD_KEY                         00000000 RG     12
CMD_LINE                        0000225C R      03
CMD_LINE_DSCR                   000020D3 R      03
CMD_START                       00000000 R      11
CMD_STATE                       00000000 RG     11
CMD_STATUS                      000004B9 R      03
CNTRLC_AST_DEL                  00001A2A RG     0F
CNVRT_ASCII                     000032D8 R      0F
CNX_STATE_ADRS                  0000038B R      09
COL_NUM                         0000224C R      03
COMMON_SNAP                     000030D4 R      0F
CONTINUOUS_QUAL                 00000029 R      02
CONVERT_DEC                     00002714 R      0F
CONVERT_HEX                     000026FC R      0F
CONVERT_TO_HEX                  00000E5C R      10
CON_STATE_ADRS                  00000663 R      09
COUNT                         = 00000019
COUNTER_OWNER_ADRS              00000135 R      0E
CROSS                         = 0000006E
CR_WAITS_ADRS                   00000838 R      09
CSB$B_ECOLVL                  = 00000040
CSB$B_ECOLVL_DSCR               00000330 R      09
CSB$B_REMACKLIM               = 00000033
CSB$B_REMACKLIM_DSCR            00000311 R      09
CSB$B_STATE                   = 00000043
CSB$B_STATE_DSCR                00000373 R      09
CSB$B_UNACKEDMSGS             = 00000032
CSB$B_UNACKEDMSGS_DSCR          000002F1 R      09
CSB$B_VERNUM                  = 00000041
CSB$B_WARMCDRPS               = 00000042
CSB$B_WARMCDRPS_DSCR            00000351 R      09
CSB$C_LENGTH                  = 000000AC
CSB$L_CSID                    = 0000004C
CSB$L_CSID_DSCR                 00000255 R      09
CSB$L_STATUS                  = 00000060
CSB$L_STATUS1                 = 00000060
CSB$L_STATUS1_DSCR              000003F2 R      09
CSB$L_STATUS2                 = 00000060
CSB$L_STATUS2_DSCR              00000412 R      09
CSB$L_STATUS_DSCR               00000434 R      09
CSB$M_LONG_BREAK              = 00000001
CSB$M_MEMBER                  = 00000002
CSB$M_REMOVED                 = 00000004
CSB$Q_REFTIME                 = 00000074
CSB$Q_REFTIME_DSCR              00000453 R      09
CSB$V_LONG_BREAK              = 00000000
CSB$V_MEMBER                  = 00000001
CSB$V_QF_ACTIVE               = 00000009
CSB$V_QF_SAME                 = 00000003
CSB$V_REMOVED                 = 00000002
CSB$W_ACKRSEQNM               = 00000030
CSB$W_ACKRSEQNM_DSCR            000002B2 R      09
CSB$W_LCKDIRWT                = 00000054
CSB$W_LCKDIRWT_DSCR             000002D2 R      09
CSB$W_QDVOTES                 = 00000056
CSB$W_QDVOTES_DSCR              000003D2 R      09
CSB$W_QUORUM                  = 00000052
CSB$W_QUORUM_DSCR               000003B3 R      09
CSB$W_RCVDSEQNM               = 0000002E
CSB$W_RCVDSEQNM_DSCR            00000292 R      09
CSB$W_SENDSEQNM               = 0000002C
CSB$W_SENDSEQNM_DSCR            00000272 R      09
CSB$W_VOTES                   = 00000050
CSB$W_VOTES_DSCR                00000395 R      09
CSB_BOF_PNTR                    0000001B R      10
CSB_COUNT                       000003E7 R      03
CSB_DSCR                        0000002F R      10
CSB_IN                        = 00000003
CSB_SNAP                        00002EB6 R      0F
CSB_SNAP_END                    00002EE0 R      0F
CSB_STATE_STRNG                 0000040B R      02
CSB_STATE_STRNG_LEN           = 0000000A
CSB_STATUS_STRNG                00000452 R      02
```

```
CSB_STATUS_STRNG_LEN            = 00000007
CSI                             = 0000009B
CSID_ADRS                         0000026D R      09
CTRLQ                           = 00000017
CTRLZ                           = 0000001A
CURRENT_CLASS                     000021F4 R      03
CURRENT_REPORT                    00000000 R      10
CUR_PRJ                           00002230 R      03
CVT_TABLE                         000004A9 R      02
DATA_BUF                          000019A3 R      03
DATA_BUF_LEN                    = 00000400
DATA_CNTRL                        000021F8 R      03
DATA_DSCR                         0000199B R      03
DC$_TERM                        = 00000042
DCRT_DWN_ARROW                  = 0000425B
DCRT_LFT_ARROW                  = 0000435B
DCRT_NARROW_DSCR                  0000014E R      02
DCRT_NOATR_DSCR                   00000143 R      02
DCRT_RT_ARROW                   = 0000445B
DCRT_UP_ARROW                   = 0000415B
DCRT_WIDE_DSCR                    0000015B R      02
DDB$C_UCB                       = 00000004
DDB$T_NAME                      = 00000014
DDB_LOOP                          000030F4 R      0F
DDM_SIZE                          0000C0D1 R      03
DEC                             = 00000003
DECIMAL_ID                        00000E93 R      10
DEC_DISPLAY                       00002653 R      0F
DEC_H_SCROLL                      0000188A R      0F
DEC_SIZE                        = 00000003
DEC_TAB                           000004CA R      02
DEC_TAB_END                       0000056A R      02
DEC_TAB_LEN                     = 00000014
DEC_V_SCROLL                      00001748 R      0F
DEF_CONNCNT                     = 00000012
DEF_EXT                           00000001 R      02
DEF_EXT_LEN                     = 00000004
DEF_NAME2                         00000005 R      02
DEF_NAME2_LEN                   = 00000016
DELAY                             000021AC R      03
DELM                              0000241F R      0F
DELM_DSCR                         00000173 R      02
DELM_DSCR_LEN                   = 00000003
DEV$V_NET                         ********    X   0F
DEV$V_TRM                         ********    X   0F
DEVDEPEND                         00002218 R      03
DEVDEPEND2                        0000221C R      03
DEVICE_CLASS                      00002208 R      03
DEVICE_TYPE                       0000220C R      03
DEV_TYP_STRNG                     000002F2 R      02
DEV_TYP_STRNG_LEN               = 00000005
DGI_MAP_ADRS                      000002BA R      0E
DGS_DSCRD_ADRS                    0000072D R      09
DGS_FREE_ADRS                     0000070F R      0E
DGS_RCVD_ADRS                     0000070C R      09
DGS_SENT_ADRS                     000006EB R      09
DG_OVRHD_SIZE_ADRS                000001A5 R      0E
DG_SIZE_ADRS                      000001E6 R      09
DIR_WT_ADRS                       000002EA R      09
DISPLAY                           00000FF9 R      0F
DISPLAY_CABLE_STATUS              00002BB2 R      0F
DISPLAY_CIR_STATE                 00002B68 R      0F
DISPLAY_CLUB                      000010A6 R      0F
DISPLAY_COMMON                    0000125A R      0F
DISPLAY_CON_STATE                 00002C21 R      0F
DISPLAY_CON_STATE_LEN           = 00000009
DISPLAY_CSB_STATE                 00002AD0 R      0F
DISPLAY_CSB_STATUS                00002AF2 R      0F
DISPLAY_END                       00001245 R      0F
DISPLAY_ERROR                     00001255 R      0F
DISPLAY_FIELD                     00002989 R      0F
DISPLAY_KBYTES                    00002C6F R      0F
DISPLAY_LOOPBACK_STATUS           00002CE9 R      0F
DISPLAY_PORT_STATUS               00002C91 R      0F
DISPLAY_PORT_TYPE                 00002CB8 R      0F
DISPLAY_PROC_NAME                 00002BF4 R      0F
DISPLAY_PROTOCOL                  00002A96 R      0F
DISPLAY_PTP                       00001170 R      0F
DISPLAY_QF                        00002A44 R      0F
DISPLAY_QF_ACTIVE                 00002A2E R      0F
DISPLAY_QF_SAME                   00002A18 R      0F
DISPLAY_QF_VOTE                   00002A02 R      0F
DISPLAY_RPORT_TYP                 00002B19 R      0F
DISPLAY_RSTATE                    00002B92 R      0F
DISPLAY_SCS                       00001109 R      0F
DISPLAY_SCS_STATE                 00002C48 R      0F
DISPLAY_SCS_STATE_LEN           = 00000009
DISPLAY_SOFTWARE                  00002A5D R      0F
DISPLAY_STRING                    000029E6 R      0F
DO_SNAPSHOT                       0000052E R      0F
DRAW_LAST_LINE                    000020F1 R      0F
DRAW_LINE                         00002466 R      0F
DT$_CI750                       = 00000002
DT$_CI780                       = 00000001
DT$_VT5X                        = 00000040
DUMMY_PB                          00000253 R      10
DUMMY_SB                          000001F3 R      10
DVI$_DEVBUFSIZ                  = 00000008
DVI$_DEVCLASS                   = 00000004
DVI$_DEVDEPEND                  = 0000000A
DVI$_DEVDEPEND2                 = 0000001C
DVI$_DEVTYPE                    = 00000006
DVT$M_AVO                       = 00000010
DVT$M_DCRT                      = 00000002
DVT$M_DISK                      = 00000020
DVT$M_HARD                      = 00000004
DVT$M_UNKNOWN                   = 00000008
DVT$M_VT52                      = 00000001
DVT$V_AVO                       = 00000004
DVT$V_DCRT                      = 00000001
DVT$V_DISK                      = 00000005
DVT$V_HARD                      = 00000002
DVT$V_UNKNOWN                   = 00000003
DVT$V_VT52                      = 00000000
```

```
ENDING_QUAL                     00000059 R      02
ERASE_LINE                      00002402 R      0F
ERASE_PAGE                      00001A99 R      0F
ERASE_TWO_LINES                 000023FD R      0F
ERR_MSGVEC                      0000049D R      03
ERR_MSG_DSCR                    00001E7E R      03
ERT_COUNT_ADRS                  0000031C R      0E
ERT_MAX_ADRS                    0000033E R      0E
ESC                         =   0000001B
EVF$M_AST_FLAG              =   00000004
EVF$M_TIM_FLAG              =   00000002
EVF$V_AST_FLAG              =   00000002
EVF$V_TIM_FLAG              =   00000001
EXCLUDE_CLASS                   00000C81 RG     10
EXE$GL_LOCKRTRY                 ******** X      0F
EXECUTE                         00001215 RG     10
EXIT_BLK                        00000481 R      03
EXIT_RTN                        00000F39 RG     0F
FAB$B_DNS                   =   0000003[illegible]
FAB$C_BID                   =   0000000[illegible]
FAB$C_BLN                   =   00000050
FAB$C_SEQ                   =   00000000
FAB$C_VAR                   =   00000002
FAB$L_ALQ                   =   00000010
FAB$L_DEV                   =   00000040
FAB$L_DNA                   =   00000030
FAB$L_FOP                   =   00000004
FAB$V_CHAN_MODE             =   00000002
FAB$V_FILE_MODE             =   00000004
FAB$V_LNM_MODE              =   00000000
FAB$V_NAM                   =   00000018
FAB$V_PUT                   =   00000000
FAB$W_GBC                   =   00000048
FD$B_FIELD_SIZE                 00000000
FD$B_FORMAT                     00000005
FD$B_LENGTH                     00000004
FD$B_MAX_DATA_SIZE              00000002
FD$B_MAX_FLD_SIZE               00000003
FD$B_MIN_FLD_SIZE               00000001
FD$B_TITLE_SIZE                 00000018
FD$L_CLASS                      0000000C
FD$L_DISPLAY_ROUTINE            00000008
FD$L_OFFSET                     00000010
FD$L_OFFSET_PRFX                00000014
FD$T_TITLE                      00000019
FDONE                           000026C7 R      0F
FERR                            000026CA R      0F
FIELD_BOT_BOX_BUF               00000B7B R      03
FIELD_BOT_BOX_DSCR              00000B73 R      03
FIELD_BOT_BOX_LEN           =   00000200
FIELD_TOP_BOX_BUF               00000973 R      03
FIELD_TOP_BOX_DSCR              0000096B R      03
FIELD_TOP_BOX_LEN           =   00000200
FIND_ENTRY                      00002363 R      0F
FIND_NODE                       00000D5B R      10
FIND_SYS_ID                     00000E16 R      10
FIRST_DATA_LINE                 00002254 R      03
FIRST_LINE                      00002250 R      03
FIX_BOF                         0000004B R      10
FIX_BUF_LEN                 =   00000458
FLAG_STATE                      00000472 R      03
FORMAT_BUF                      0000215B R      03
FORMAT_BUF_DSCR                 00002153 R      03
FORMAT_BUF_LEN                  00002153 R      03
FORMAT_DATA                     0000257F R      0F
FORMED_ADRS                     000000DD R      09
FORM_CIRCS_ADRS                 0000020E R      0E
FORM_PB_DONE                    000031D2 R      0F
FORM_PB_LOOP                    00003140 R      0F
FORM_PB_SNAP                    0000311E R      0F
FREE_BUFF_DESCRIPTORS_ADRS      00000064 R      0E
FREE_BUFF_MSG                   00000205 R      02
FREE_BUF_END                    000032B8 R      0F
FREE_BUF_LOOP                   000032AE R      0F
FREE_BUF_SNAP                   0000329E R      0F
GET_BLANK                       00002520 R      0F
GET_CLASSES                     000022A0 R      0F
GET_DATA                        00002729 RG     0F
GET_HEADER                      00001C46 R      0F
GET_HELP                        00001077 RG     10
GET_LIST                        0000057E R      02
GET_MSG_BUF                     00001E8E R      03
GET_MSG_BUF_LEN             =   00000100
GET_MSG_DSCR                    00001E86 R      03
GET_NODE_NAME                   00000F1B RG     0F
GET_NUM_PB                      000008FB R      10
GET_TITLES                      00002144 R      0F
GET_VAL_DSCR                    00001DA3 R      03
GRAPHICS_DSGNTR                 00000168 R      02
GRAPHICS_OFF                =   0000000F
GRAPHICS_ON                 =   0000000E
HEADER_END                  =   00000007
HEADER_SIZE                     000021C4 R      03
HEADER_START                =   00000002
HELP                            0000064A R      11
HELP_DSCR                       00000553 R      03
HELP_FLAGS                      0000047D R      03
HEX                         =   00000002
HEX_DISPLAY                     000025F9 R      0F
HIGH_NYBLE                      0000262C R      0F
HLP$M_HELP                  =   00000020
HLP$M_PROMPT                =   00000001
HW_TYPE_ADRS                    00000180 R      09
HW_TYPE_TABLE                   000004B2 R      02
HW_VERS_ADRS                    00000160 R      09
H_SCROLL_LINE                   0000046C R      03
ID_COUNT                        000022D8 R      03
ID_LIST_ENTRYS              =   00000010
ID_LIST_LEN                 =   00000060
ID_RESULT                       000022DC R      03
INAM_BLOCK                      0000006C R      03
INCARNATION_ADRS                00000227 R      09
INCLUDE_CLASS                   00000C2D RG     10
INCN_TIME_ADRS                  0000024B R      09
```

```
INC_H_SCROLL                    0000185D R      0F
INC_V_SCROLL                    000017C5 R      0F
INFAB                           0000001C R      03
INIT                            00000862 R      0F
INIT_NAME                       0000012A R      02
INIT_REC_ADRS                   000008FA R      09
INRAB                           000002CC R      03
INSTRNG_LEN                     000004C5 R      03
INTERLOCK_SNAP                  00003249 R      0F
INTERVAL_QUAL                   0000003B R      02
INVRS_DSCR                      0000017E R      02
IN_CHAN                         00002234 R      03
IN_DEV                          00002246 R      03
IN_DEV_NAME                     000000DE R      02
IN_EXP_DSCR                     0000011C R      03
IN_EXP_NAME                     00000124 R      03
IN_EXP_NAME_LEN               = 00000050
IN_FILE_SPEC                    000000CC R      03
IN_FILE_SPEC_LEN              = 00000050
IO$M_CTRLCAST                   ********   X    0F
IO$M_DSABLMBX                   ********   X    0F
IO$M_ENABLMBX                   ********   X    0F
IO$M_EXTEND                     ********   X    0F
IO$M_TYPEAHDCNT                 ********   X    0F
IO$_READPROMPT                  ********   X    0F
IO$_READVBLK                    ********   X    0F
IO$_SENSEMODE                   ********   X    0F
IO$_SETMODE                     ********   X    0F
IO$_WRITEVBLK                   ********   X    0F
IOC$GL_DEVLIST                  ********   X    0F
IOSB                            0000223E R      03
IPL$_SCS                      = 00000008
JPI$_CURPRIV                    ********   X    02
JPI_LIST                        000005BE R      02
KB_MAPPED_ADRS                  00000816 R      09
KB_RCVD_ADRS                    000007F6 R      09
KB_SENT_ADRS                    000007D6 R      09
KEYTAB_ADRS                     00002274 R      03
KEYWRDVAL                     = 00000019
KEY_DSCR                        0000054B R      03
KEY_MSK                         00000547 R      03
KEY_OFF                         00000543 R      03
KEY_VAL                         0000053F R      03
LAST_BRDCST_BUF                 00001DFA R      03
LAST_BRDCST_DSCR                00001DF2 R      03
LAST_CLUB_LINE_BUF              00000F8B R      03
LAST_CLUB_LINE_DSCR             00000F83 R      03
LAST_CLUB_LINE_LEN            = 00000200
LAST_DATA_LINE                  00002258 R      03
LAST_LINE                       00002260 R      03
LAST_LINE_BUF                   00000D83 R      03
LAST_LINE_DSCR                  00000D7B R      03
LAST_LINE_LEN                 = 00000200
LAST_TRANSITION_ADRS            000000FC R      09
LBR$OUTPUT_HELP                 ********   X    10
LB_STATUS_ADRS                  000002FA R      0E
LB_STAT_STRG                    000002AC R      02
LB_STAT_STRG_LEN              = 00000005
LEFT_T                        = 00000074
LEN                           = 00000009
LIB$ASN_WTH_MBX                 ********   X    0F
LIB$CVT_DTB                     ********   X    0F
LIB$CVT_TIME                    ********   X    0F
LIB$DATE_TIME                   ********   X    0F
LIB$DOWN_SCROLL                 ********   X    0F
LIB$ERASE_LINE                  ********   X    0F
LIB$ERASE_PAGE                  ********   X    0F
LIB$FREE_VM                     ********   X    0F
LIB$GET_INPUT                   ********   X    0F
LIB$GET_VM                      ********   X    0F
LIB$LOOKUP_KEY                  ********   X    10
LIB$PUT_BUFFER                  ********   X    0F
LIB$PUT_OUTPUT                  ********   X    10
LIB$PUT_SCREEN                  ********   X    0F
LIB$SET_BUFFER                  ********   X    0F
LIB$SET_CURSOR                  ********   X    0F
LIB$SET_OUTPUT                  ********   X    0F
LIB$SET_SCROLL                  ********   X    0F
LIB$SIGNAL                      ********   X    0F
LIB$TPARSE                      ********   X    0F
LIB$UP_SCROLL                   ********   X    0F
LIB$_AMBKEY                     ********   X    0F
LIB$_SYNTAXERR                  ********   X    0F
LIB$_UNRKEY                     ********   X    0F
LIBRARY_NAME                    0000010C R      02
LINE_DIF                        00000434 R      03
LINE_NUM                        00002248 R      03
LISTENER_SNAP                   00003030 R      0F
LISTENER_SNAP_END               000030CD R      0F
LL_CORNER                     = 0000006D
LOCAL_BITMAP                    00000000 R      0B
LOCAL_H_SCROLL                  00000468 R      03
LOCAL_KEYTAB                    00000000 R      0D
LOCAL_MAP_LEN                 = 00000000
LOCAL_MSG                       000001A6 R      02
LOCAL_REPORT                    00001513 R      0F
LOCAL_SAVE                      00002200 R      03
LOCAL_SCROLL_FLAGS              0000047B R      03
LOCAL_SIZE_ARRAY                00000000 R      0A
LOCAL_SNAP                      000030DF R      0F
LOCAL_TABLE                     00000000 R      0C
LOCAL_TAB_LEN                 = 00000064
LOCK_FAIL                       0000332F R      0F
LOCSYSID                        00000005 R      10
LOCSYSID_DSCR                   00000000 R      0E
LOC_CONID_ADRS                  00000685 R      09
LOC_PROC_NAME_ADRS              00000617 R      09
LOC_SAVE_BITMAP                 00000004 R      0B
LOC_SB_ADRS                     00000013 R      10
LOG_MAP_ADRS                    000002DA R      0E
LOOKUP_FIELD                    00000B3A RG     10
LOOP                            00003318 R      0F
LOWERIPL                        0000332C R      0F
LOW_NYBLE                       00002638 R      0F
```

```
LPORT_NAME_ADRS                 00000493 R      09
LP_STATUS_ADRS                  000000AF R      0E
LP_TYPE_ADRS                    000001CB R      0E
LR_CORNER                     = 0000006A
LSYSTEM_ID_ADRS                 00000018 R      0E
MAIN_CONT                       0000018D R      0F
MAIN_ERR                        00000F42 R      0F
MAIN_EXIT                       00000FE8 R      0F
MAIN_LOOP                       00000012 R      0F
MAP_LEN                         0000226C R      03
MAP_LENGTH                    = 00000000
MASK                          = 01C060F8
MASK_BIT                      = 02000000
MAXPDT                        = 00000002
MAX_ID                        = 00000010
MAX_MSG                         000004B1 R      03
MAX_PORT_ADRS                   00000279 R      0E
MB$B_ERTCNT                     00000010
MB$B_ERTCNT_DSCR                00000304 R      0E
MB$B_ERTMAX                     00000011
MB$B_ERTMAX_DSCR                00000326 R      0E
MB$L_DEVTYPE                    00000008
MB$L_DEVTYPE_DSCR               000001B3 R      0E
MB$L_FORM_PB                    00000014
MB$L_FORM_PB_DSCR               000001F6 R      0E
MB$L_PL_WAITERS                 00000018
MB$L_PL_WAITERS_DSCR            00000143 R      0E
MB$L_PORT_NAME                  00000000
MB$L_PORT_NAME_DSCR             0000007A R      0E
MB$L_PORT_NUM                   00000004
MB$L_PORT_NUM_DSCR              000000B9 R      0E
MB$L_STS                        0000000C
MB$L_STS_DSCR                   00000097 R      0E
MB$W_CMDS_QUED                  00000020
MB$W_CMDS_QUED_DSCR             00000219 R      0E
MB$W_DG_FREE                    0000001C
MB$W_DG_FREE_DSCR               000000DA R      0E
MB$W_ERRCNT                     00000012
MB$W_ERRCNT_DSCR                00000346 R      0E
MB$W_MSG_FREE                   0000001E
MB$W_MSG_FREE_DSCR              000000FB R      0E
MB$W_RSPS_QUED                  00000022
MB$W_RSPS_QUED_DSCR             0000023D R      0E
MBXCHAN                         0000223C R      03
MBX_BUF                         00001FB5 R      03
MBX_BUF_LEN                   = 00000100
MBX_MSG_DSCR                    00001FAD R      03
MB_IN                         = 00000007
MESEQ_ADRS                      000000BE R      09
MESSAGE                         00002FF8 R      0F
MF$M_CONT                     = 00000010
MF$M_CR_FOUND                 = 00000400
MF$M_CYCLE                    = 00000020
MF$M_EXIT_FLAG                = 00000002
MF$M_INIT_FLAG                = 00000001
MF$M_INVERSE                  = 00000004
MF$M_IN_OPEN                  = 00000040
MF$M_NEW_CHAR                 = 00000080
MF$M_RIP                      = 00000008
MF$M_SCROLL                   = 00000100
MF$M_VALID_MSG                = 00000200
MF$V_CONT                     = 00000004
MF$V_CR_FOUND                 = 0000000A
MF$V_CYCLE                    = 00000005
MF$V_EXIT_FLAG                = 00000001
MF$V_INIT_FLAG                = 00000000
MF$V_INVERSE                  = 00000002
MF$V_IN_OPEN                  = 00000006
MF$V_NEW_CHAR                 = 00000007
MF$V_RIP                      = 00000003
MF$V_SCROLL                   = 00000008
MF$V_VALID_MSG                = 00000009
MIN_REC_ADRS                    0000091B R      09
MIN_SEND_ADRS                   00000898 R      09
MISC_BUF                        00000447 R      10
MISC_BUF_LEN                  = 0000002E
MISC_FLAGS                      00000476 R      03
MORE_DSCR                       00000078 R      02
MORE_DWN                        000000B2 R      02
MORE_LFT                        000000BD R      02
MORE_RT                         000000C8 R      02
MORE_TABS                       00000090 R      02
MORE_UP                         000000A7 R      02
MSG$_TRMBRDCST                = 00000053
MSG$_TRMUNSOLIC               = 00000001
MSG1_DSCR                       000001BC R      02
MSG2_DSCR                       000001CF R      02
MSGS_FREE_ADRS                  00000113 R      0E
MSGS_RCVD_ADRS                  00000771 R      09
MSGS_SENT_ADRS                  0000074F R      09
MSG_FRE_QUE                     00000008 R      03
MSG_HDR_SIZE_ADRS               0C000180 R      0E
MSG_OUT_ADRS                    000004AD R      03
MSG_OUT_QUE                     00000010 R      03
MSG_SIZE_ADRS                   00000206 R      09
NAM$B_ESS                     = 0000000A
NAM$B_NOP                     = 00000008
NAM$B_RSS                     = 00000002
NAM$C_BID                     = 00000002
NAM$C_BLN                     = 00000060
NAM$L_ESA                     = 0000000C
NAM$L_RSA                     = 00000004
NAM$T_DVI                     = 00000014
NAME_ADRS                       00000092 R      0E
NARROW                        = 00000050
NEW_CHAR_BUF                    00002220 R      03
NEXT_ADD_CIR_QUAL               00000286 R      11
NEXT_ADD_CON_QUAL               000002F9 R      11
NEXT_ADD_SYS_QUAL               0000019C R      11
NEXT_REM_CIR_QUAL               00000509 R      11
NEXT_REM_CON_QUAL               0000056F R      11
NEXT_REM_SYS_QUAL               0000042C R      11
NEXT_SET_FIELD_QUAL             000005D8 R      11
NODE_ADRS                       00000124 R      09
```

```
NODE_NAME                       00000450 R      03
NODE_NAME_DSCR                  00000448 R      03
NO_MESSAGE                      00003001 R      0F
NO_MORE                         000000D3 R      02
NUCL_DEV_NAME                   00000101 R      02
NUM_BUF_DSCR                    0000000B R      10
NUM_BUF_DSCR_DSCR               00000023 R      0E
NUM_CIRCUITS_ADRS               000001C1 R      09
NUM_CLASSES                   = 00000009
NUM_CLUSTER_DSCRS             = 00000046
NUM_CONNECTIONS_ADRS            00000519 R      09
NUM_ERRORS_ADRS                 0000035E R      0E
NUM_FIELDS                      00000425 R      03
NUM_FREE_BUF_DSCR               0000000F R      10
NUM_FREE_BUF_DSCR_DSCR          0000004C R      0E
NUM_HW_TYPES                  = 00000006
NUM_LINES                       00000428 R      03
NUM_LOCAL_DSCRS               = 00000019
OLD_CDT_DSCR                    00000411 R      03
OLD_CSB_DSCR                    00000401 R      03
OLD_FIX_BUF                     000004A3 R      10
OLD_PB_DSCR                     00000409 R      03
OLD_SB_DSCR                     000003F9 R      03
OLD_VAR_BUF                     00000419 R      03
OLD_VAR_BUF_END                 0000041D R      03
OLD_VAR_BUF_LEN                 00000421 R      03
OPEN_CIRCS_ADRS                 000001EB R      0E
OTS$CVT_TZ_L                    ******** X      10
OUTFAB                          00000174 R      03
OUTNAM_BLOCK                    000001C4 R      03
OUTPUT_DDM                      00001D97 R      0F
OUTPUT_QUAL                     0000004B R      02
OUTPUT_TIME                     00001E62 R      0F
OUTPUT_WIDTH                    0000222C R      03
OUT_CHAN                        00002236 R      03
OUT_DEV                         00002247 R      03
OUT_DEV_NAME                    000000EF R      02
OUT_EXP_DSCR                    00000274 R      03
OUT_EXP_NAME                    0000027C R      03
OUT_EXP_NAME_LEN              = 00000050
OUT_FILE_SPEC                   00000224 R      03
OUT_FILE_SPEC_LEN             = 00000050
OUT_MSG                         0000161A R      0F
PAAC                          = 41415003
PABC                          = 42415003
PARSE_INTERVAL                  00000624 R      11
PARSE_SCREEN                    0000062E R      11
PARSE_VALUE                     00000638 R      11
PB$B_CBL_STS                  = 00000028
PB$B_CBL_STS_DSCR               0000056D R      09
PB$B_PO_STS                   = 00000029
PB$B_P1_STS                   = 0000002A
PB$B_RSTATE                   = 00000021
PB$B_RSTATE_DSCR                0000054B R      09
PB$B_RSTATION                 = 0000000C
PB$B_RSTATION_DSCR              0000049E R      09
PB$B_RST_PORT                 = 00000020
PB$B_RST_PORT_DSCR              000004C0 R      09
PB$C_LENGTH                   = 00000054
PB$C_OPEN                     = 00000003
PB$C_VC_FAIL                  = 00008000
PB$L_BLINK                    = 00000004
PB$L_CDTLST                   = 00000034
PB$L_RPORT_FCN                = 0000001C
PB$L_RPORT_FCN_DSCR             000005B6 R      09
PB$L_RPORT_REV                = 00000018
PB$L_RPORT_REV_DSCR             00000592 R      09
PB$L_RPORT_TYP                = 00000014
PB$L_RPORT_TYP_DSCR             000004E1 R      09
PB$L_SBLINK                   = 00000030
PB$L_WAITQFL                  = 00000038
PB$L_WAITQFL_DSCR               000005DB R      09
PB$M_CUR_CBL                  = 00000001
PB$M_CUR_PS                   = 00000001
PB$SRC$L_NUM_CDT              = 00000004
PB$SHC$L_NUM_CDT_DSCR           00000501 R      09
PB$S_PORT_TYP                 = 0000001F
PB$T_LPORT_NAME               = 00000024
PB$T_LPORT_NAME_DSCR            0000047B R      09
PB$V_PORT_TYP                 = 00000000
PB$W_STATE                    = 00000012
PB$W_STATE_DSCR                 00000529 R      09
PB_BUF_PNTR                     0000001F R      10
PB_COUNT                        000003E9 R      03
PB_DSCR                         00000037 R      10
PB_FIRST_LOOP                   00002F25 R      0F
PB_IN                         = 00000004
PB_LOOP                         00002F0B R      0F
PB_LOOP_END                     0000301B R      0F
PB_RSTATE_STRNG                 00000330 R      02
PB_RSTATE_STRNG_LEN           = 00000008
PB_SNAP                         00002EE4 R      0F
PB_STATE_STRNG                  0000030D R      02
PB_STATE_STRNG_LEN            = 00000007
PDT$B_DQIMAP                    00000154
PDT$B_DQIMAP_DSCR               000002A2 R      0E
PDT$B_HSHUT_DG                  000001B0
PDT$B_MAX_PORT                  0000017C
PDT$B_MAX_PORT_DSCR             00000261 R      0E
PDT$B_NXT_PORT                  0000017E
PDT$B_PO_LBSTS                  00000180
PDT$B_PO_LBSTS_DSCR             000002E2 R      0E
PDT$B_P1_LBSTS                  00000181
PDT$B_PLOGMAP                   00000134
PDT$B_PLOGMAP_DSCR              000002C2 R      0E
PDT$B_PORTMAP                   00000114
PDT$B_PORTMAP_DSCR              00000282 R      0E
PDT$B_PORT_NUM                  0000017D
PDT$B_REQIDPS                   0000017F
PDT$C_LENGTH                  = 000000E4
PDT$C_PAREGBASE                 000000E4
PDT$C_PAREGEND                  00000110
PDT$C_PQB                     = 000001E0
PDT$L_CNF                       000000E4
```

```
PDT$L_CQO                       000000F0
PDT$L_CQ1                       000000F4
PDT$L_DFQ                       000000FC
PDT$L_DFQHDR                    00000208
PDT$L_DGHDRSZ                   00000190
PDT$L_DGNETHD                   00000194
PDT$L_DGOVRHD                 = 000000B8
PDT$L_DGOVRHD_DSCR              0000018D R      0E
PDT$L_DQELOGOUT                 000002E0
PDT$L_GPTBASE                   0000022C
PDT$L_GPTLEN                    00000230
PDT$L_LBDG                      00000184
PDT$L_MFQ                       00000100
PDT$L_MFQHDR                    0000020C
PDT$L_MQELOGOUT                 00000320
PDT$L_MSGHDRSZ                = 000000B4
PDT$L_MSGHDRSZ_DSCR             00000168 R      0E
PDT$L_MTC                       00000104
PDT$L_PFAR                      00000108
PDT$L_PMC                       000000E8
PDT$L_POLLERDUE                 0000018C
PDT$L_POOLDUE                   00000188
PDT$L_PPR                       0000010C
PDT$L_PS                        000000EC
PDT$L_PSR                       000000F8
PDT$L_SPTBASE                   00000224
PDT$L_SPTLEN                    00000228
PDT$L_VBDT                      0000021C
PDT$L_VPQB                      00000218
PDT$L_WAITQFL                 = 000000AC
PDT$M_PRV_LBS                 = 00000002
PDT$Q_COMQ2                     000001F0
PDT$Q_COMQ3                     000001F8
PDT$Q_COMQBASE                  000001E0
PDT$Q_COMQH                     000001E8
PDT$Q_COMQL                     000001E0
PDT$Q_DFREEQ                    000001D0
PDT$Q_FORMPB                    00000174
PDT$Q_MFREEQ                    000001D8
PDT$Q_RSPQ                      00000200
PDT$Q_TEMP_RSPQ                 0000019C
PDT$T_CNTOWNER                = 000000C4
PDT$T_CNTOWNER_DSCR             0000011D R      0E
PDT$V_LBDG                    = 00000002
PDT$W_BDTLEN                    00000220
PDT$W_DQELEN                    00000210
PDT$W_LPORT_STS                 00000110
PDT$W_MQELEN                    00000214
PDT$W_PBCOUNT                   00000112
PDT$W_PBCOUNT_DSCR              000001D3 R      0E
PDT$W_SIZE                    = 00000008
PDT$W_STDGDYN                   00000198
PDT$W_STDGUSED                  0000019A
PDT_BOF                         000002A7 R      10
PDT_IN                        = 00000006
PDT_PFX                         000004A0 R      02
PEND_REC_ADRS                   000008D9 R      09
PFX_LIM                       = 00000007
POOL_WAITERS_ADRS               0000015B R      0E
POOL_WAITER_LOOP                0000323C R      0F
POOL_WAITER_LOOP_END            00003249 R      0F
PORT_FORM_PB_LOOP               0000320F R      0F
PORT_FORM_PB_LOOP_END           0000321C R      0F
PORT_NUM_ADRS                   000000D1 R      0E
PORT_SNAP                       000031D9 R      0F
PORT_SNAP_END                   000032D1 R      0F
PORT_STATUS_STRNG               000002FE R      02
PORT_STATUS_STRNG_LEN         = 00000007
PPD$B_DEF_ST                    0000001C
PPD$B_FLAGS                     0000000F
PPD$B_HWVERS                    00000034
PPD$B_LBDATA                    00000012
PPD$B_LCB_0                     00000012
PPD$B_LCB_LPORT                 00000010
PPD$B_LCB_NPORT                 0000000F
PPD$B_LCB_OPC                   00000011
PPD$B_LCB_PORT                  0000000E
PPD$B_OPC                       0000000E
PPD$B_PORT                      0000000C
PPD$B_PROTOCOL                  0000001A
PPD$B_RSTATE                    00000025
PPD$B_RST_PORT                  00000024
PPD$B_STATUS                    0000000D
PPD$B_SWFLAG                    0000000B
PPD$B_SYSTEMID                  00000014
PPD$B_TYPE                      0000000A
PPD$C_LB_LENGTH                 00000046
PPD$C_LCB_DATA                  00000013
PPD$C_LENGTH                    00000012
PPD$C_MIN_DGSIZ                 00000050
PPD$K_LB_LENGTH                 00000046
PPD$K_LENGTH                    00000012
PPD$L_BLINK                     00000004
PPD$L_DG_DISC                   00000028
PPD$L_FLINK                     00000000
PPD$L_IN_VCD                    00000018
PPD$L_LBCRC                     00000042
PPD$L_PO_ACK                    00000010
PPD$L_PO_NAK                    00000014
PPD$L_PO_NRSP                   00000018
PPD$L_P1_ACK                    0000001C
PPD$L_P1_NAK                    00000020
PPD$L_P1_NRSP                   00000024
PPD$L_REC_BOFF                  00000028
PPD$L_REC_NAME                  00000024
PPD$L_RPORT_FCN                 00000020
PPD$L_RPORT_REV                 0000001C
PPD$L_RPORT_TYP                 00000018
PPD$L_SND_BOFF                  00000020
PPD$L_SND_NAME                  0000001C
PPD$L_ST_ADDR                   00000018
PPD$L_XCT_LEN                   00000018
PPD$Q_CURTIME                   00000048
PPD$Q_NODENAME                  00000040
```

```
PPD$Q_SWINCARN                  00000028
PPD$Q_XCT_ID                    00000010
PPD$T_HWTYPE                    00000030
PPD$T_SWTYPE                    00000020
PPD$T_SWVERS                    00000024
PPD$W_LCB_LEN7                  0000000C
PPD$W_LENGTH                    00000010
PPD$W_MASK                      00000010
PPD$W_MAXDG                     0000001C
PPD$W_MAXMSG                    0000001E
PPD$W_MTYPE                     00000012
PPD$W_M_VAL                     00000014
PPD$W_SIZE                      00000008
PR$_IPL                         ********    X   0F
PRIOR_BLANK_LINES               00000440 R      03
PRIOR_LINE_DIF                  00000438 R      03
PRIOR_NUM_LINES                 0000042C R      03
PROMPT                          00000229 R      02
PROMPT_LEN                    = 0000000A
PROTOCOL_ADRS                   00000348 R      09
PRT_MAP_ADRS                    0000029A R      0E
PRV$V_CMKRNL                  = 00000000
QDVOTES_ADRS                    000003EA R      09
QD_NAME_ADRS                    0000005B R      09
QF_ACTIVE_ADRS                  0000042A R      09
QF_SAME_ADRS                    0000040A R      09
QF_VOTE_ADRS                    0000007B R      09
QUE_LIMIT                     = 00000003
QUORUM_ADRS                     000003CB R      09
RAB$B_RAC                     = 0000001E
RAB$C_BID                     = 00000001
RAB$C_BLN                     = 00000044
RAB$C_SEQ                     = 00000000
RAB$L_CTX                     = 00000018
RAB$L_RBF                     = 00000028
RAB$L_ROP                     = 00000004
RAB$W_RSZ                     = 00000022
RCVD_SQ_ADRS                    000002AA R      09
READ_LOOP                       00000199 R      0F
READ_WO_PROMPT                  000001F7 R      0F
READ_W_PROMPT                   00000295 R      0F
RECEIVE_ADRS                    000008B9 R      09
REC_BUFF                        00000318 R      03
REC_DSCR                        00000310 R      03
REC_SIZE                      = 00000084
REFRESH                         00002348 RG     0F
REMOVE                          00000346 R      11
REMOVE_FIELD                    00000AC6 RG     10
REMOVE_FROM_LIST                00000DFE R      10
REM_CIR                         000004F0 R      11
REM_CIR_QUAL                    00000501 R      11
REM_CIR_TYPE                    00000511 R      11
REM_CIR_TYPE_LOOP               0000051F R      11
REM_CIR_TYPE_STRING             0000052B R      11
REM_CLUSTER                     000003BD R      11
REM_CON                         00000556 R      11
REM_CONID_ADRS                  000006A7 R      09
REM_CON_QUAL                    00000567 R      11
REM_CON_TYPE                    00000577 R      11
REM_CON_TYPE_LOOP               00000585 R      11
REM_CON_TYPE_STRING             00000591 R      11
REM_COUNTERS                    000003D7 R      11
REM_CREDITS                     000003E4 R      11
REM_ERRORS                      000003FE R      11
REM_ID                          00000D18 RG     10
REM_ID_LOOP                     00000442 R      11
REM_LOCAL_PORTS                 000003F1 R      11
REM_MEMBERS                     000003CA R      11
REM_NEXT_CLASS                  000003A1 R      11
REM_NODE                        00000CFC RG     10
REM_NODE_LOOP                   000004D6 R      11
REM_PROC_NAME_ADRS              0000063D R      09
REM_STATE_ADRS                  00000563 R      09
REM_SYS                         0000040B R      11
REM_SYS_ID                      00000434 R      11
REM_SYS_ID_STRING               0000044E R      11
REM_SYS_NODE                    000004C8 R      11
REM_SYS_NODE_STRING             000004E2 R      11
REM_SYS_QUAL                    0000041C R      11
REM_SYS_TYPE                    0000045C R      11
REM_SYS_TYPE_LOOP               0000046A R      11
REM_SYS_TYPE_STRING             00000476 R      11
REPORT_QUAL                     0000001B R      02
RESET                           00001129 RG     10
RET_BUF                         00001DB3 R      03
RET_BUF_DSCR                    00001DAB R      03
RET_BUF_LEN                   = 0000003F
RIGHT                         = 00000075
RMS$_DEV                        ********    X   10
RMS$_EOF                        ********    X   10
RPORT_NUM_ADRS                  000004B6 R      09
RPORT_TYP_STRNG                 000002CA R      02
RPORT_TYP_STRNG_LEN           = 00000006
RPT$M_CLSTR                   = 00000001
RPT$V_CLSTR                   = 00000000
RP_FUNCTIONS_ADRS               000005CE R      09
RP_OWNER_ADRS                   000004D8 R      09
RP_REVISION_ADRS                000005AA R      09
RP_TYPE_ADRS                    000004F9 R      09
RSPS_QUEUED_ADRS                00000255 R      0E
SAVED_CLASSES                   000021FC R      03
SAVE_CHAR_BUF                   00002214 R      03
SAVE_OLD                        00000835 R      0F
SAVE_TITLE_LEN                  00002252 R      0F
SB$B_HWVERS                   = 00000038
SB$B_HWVERS_DSCR                00000148 R      09
SB$B_SYSTEMID                 = 00000018
SB$B_SYSTEMID_DSCR              00000129 R      09
SB$C_LENGTH                   = 00000060
SB$L_BLINK                    = 00000004
SB$L_CSB                      = 0000005C
SB$L_FLINK                    = 00000000
SB$L_PBFL                     = 0000000C
SB$Q_SWINCARN                 = 0000002C
```

```
SB$Q_SWINCARN_2                 = 0000002C
SB$Q_SWINCARN_2_DSCR              00000233 R      09
SB$Q_SWINCARN_DSCR                0000020F R      09
SB$SHC$L_NUM_PB                 = 00000004
SB$SHC$L_NUM_PB_DSCR              000001A9 R      09
SB$T_HWTYPE                     = 00000034
SB$T_HWTYPE_DSCR                  00000168 R      09
SB$T_NODENAME                   = 00000044
SB$T_NODENAME_DSCR                0000010C R      09
SB$T_SWTYPE                     = 00000024
SB$T_SWVERS                     = 00000028
SB$T_SWVERS_DSCR                  00000188 R      09
SB$W_MAXDG                      = 00000020
SB$W_MAXDG_DSCR                   000001CE R      09
SB$W_MAXMSG                     = 00000022
SB$W_MAXMSG_DSCR                  000001EE R      09
SB$W_SIZE                       = 00000008
SB_BOF_PNTR                       00000017 R      10
SB_COUNT                          000003E5 R      03
SB_DSCR                           00000027 R      10
SB_FIRST_LOOP                     00002E87 R      0F
SB_IN                           = 00000002
SB_LOOP                           00002E64 R      0F
SB_LOOP_END                       00003029 R      0F
SB_SNAP                           00002E40 R      0F
SCAN_S                          = 00000071
SCR$M_MORE                      = 00000010
SCR$M_MORE_DWN                  = 00000008
SCR$M_MORE_LFT                  = 00000002
SCR$M_MORE_RT                   = 00000001
SCR$M_MORE_UP                   = 00000004
SCR$V_MORE                      = 00000004
SCR$V_MORE_DWN                  = 00000003
SCR$V_MORE_LFT                  = 00000001
SCR$V_MORE_RT                   = 00000000
SCR$V_MORE_UP                   = 00000002
SCRATCH                           00000018 R      03
SCREEN_BUF                        00000563 R      03
SCREEN_BUF_LEN                  = 00000200
SCREEN_DSCR                       0000055B R      03
SCREEN_WIDTH                      00002222 R      03
SCROLL_DELAY                      00000458 R      03
SCROLL_DIF                        00000444 R      03
SCROLL_FLAGS                      0000047A R      03
SCROLL_LINES                      00000430 R      03
SCROLL_TWO                        00001B5F R      0F
SCS$B_CON_DAT                   = 00000024
SCS$GA_LOCALSB                    ********    X   0F
SCS$GB_SYSTEMID                   ********    X   0F
SCS$GL_BDT                        ********    X   0F
SCS$GL_CDL                        ********    X   0F
SCS$GQ_CONFIG                     ********    X   0F
SCS$GQ_DIRECT                     ********    X   0F
SCS$GW_BDTCNT                     ********    X   0F
SCS$GW_CDTCNT                     ********    X   0F
SCS$T_DST_PROC                  = 00000004
SCS$T_DST_PROC_DSCR               000005FF R      09
SCS$T_SRC_PROC                  = 00000014
SCS$T_SRC_PROC_DSCR               00000625 R      09
SCS_DIR_LOOP                      0000305A R      0F
SCS_HEADER_SIZE                   000021CC R      03
SCS_NUM_FIELDS                    00000426 R      03
SCS_REPORT                        000012B5 R      0F
SCS_STATE_ADRS                    000006C9 R      09
SCS_WAITERS_ADRS                  000005F3 R      09
SCS_WAITER_LOOP                   00002F42 R      0F
SCS_WAITER_SNAP                   00002F37 R      0F
SCS_WT_LP_END                     00002F4E R      0F
SDIR$B_PROCNAM                  = 0000000C
SDIR$L_CONID                    = 0000002C
SECOND_CHANCE                     0000040F RG     0F
SEND_ADRS                         0000087B R      09
SEND_SQ_ADRS                      0000028A R      09
SET                               000005BC R      11
SET_AST                           00001A39 R      0F
SET_CHAN                          00002238 R      03
SET_FIELD                         000005D0 R      11
SET_FIELD_FORMAT                  000005FC R      11
SET_FIELD_QUAL                    000005E0 R      11
SET_FIELD_WIDTH                   000005F2 R      11
SET_FORMAT                        00000F34 RG     10
SET_INTERVAL                      00000F98 RG     10
SET_NEXT                          00000618 R      11
SET_REGION_COMMAND                000019EA R      0F
SET_REGION_DATA                   000019DC R      0F
SET_REGION_FULL                   000019CE R      0F
SET_REGION_HELP                   00001A03 R      0F
SET_SCREEN                        00000FBA RG     10
SET_SCROLL                        000016B9 R      0F
SET_WIDTH                         00000F03 RG     10
SHC$C_CDT_LENGTH                = 0000008C
SHC$L_CDT_BEGIN                 = 00000014
SHC$L_CDT_DIF                   = FFFFFFF0
SHC$L_PDT_BEGIN                 = 000000B4
SHC$L_PDT_DIF                   = FFFFFF4C
SHC$L_PDT_LENGTH                = 000000D0
SHC$SB$B_DIS_POS                = 00000009
SHOW                              00000030 R      11
SHOW$_AMBIG                       ********    X   0F
SHOW$_CACLS                       ********    X   10
SHOW$_CRCLS                       ********    X   10
SHOW$_HIBER                       ********    X   0F
SHOW$_IVADDID                     ********    X   10
SHOW$_IVADDNODE                   ********    X   10
SHOW$_IVREMID                     ********    X   10
SHOW$_IVREMNODE                   ********    X   10
SHOW$_NOAVAIL                     ********    X   10
SHOW$_UNRCMD                      ********    X   0F
SHOW$_UNRKEY                      ********    X   0F
SHOW_CLUSTER                      00000929 RG     10
SHOW_CLUSTER$COMMAND_TABLE        ********    X   0F
SHOW_LOCAL                        0000099F RG     10
SIGNAL                            0000155D R      0F
SNAPSHOT                          00002DE6 RG     0F
```

```
SNAPSHOT_END                    000032D1 R      0F
SNAPSHOT_ERR                    000032D4 R      0F
SNAP_ADRS                       0000056A R      02
SNAP_BUF                        00000000 R      10
SNAP_BUF_END                    000004A3 R      10
SNAP_BUF_LEN                =   000004A3
SNAP_END                        00003333 R      0F
SOFTWARE_ADRS                   000001A0 R      09
SS$_BADPARAM                =   00000014
SS$_INTERLOCK               =   0000038C
SS$_NOCMKRNL                =   00002804
SS$_NORMAL                  =   00000001
SS$_NOSUCHNODE              =   0000028C
SS$_NOTRAN                  =   00000629
START                           00000000 RG     0F
START_BIT                       00002270 R      03
STATUS                          00000491 R      03
STATUS_ADRS                     0000044C R      09
STORE_MSG                       000004E9 R      0F
STR$UPCASE                      ********   X    0F
SWITCH                          000021AB R      03
SWITCH_DISPLAY                  00002566 R      0F
SYM_PRFX                        00000489 R      02
SYS$ASCTIM                      ********  GX    0F
SYS$ASSIGN                      ********  GX    0F
SYS$CANTIM                      ********  GX    0F
SYS$CLOSE                       ********  GX    10
SYS$CLREF                       ********  GX    0F
SYS$CMKRNL                      ********  GX    0F
SYS$CONNECT                     ********  GX    10
SYS$DCLEXH                      ********  GX    0F
SYS$EXIT                        ********  GX    0F
SYS$GET                         ********  GX    10
SYS$GETDVI                      ********  GX    0F
SYS$GETJPI                      ********  GX    0F
SYS$GETMSG                      ********  GX    0F
SYS$HIBER                       ********  GX    0F
SYS$LKWSET                      ********  GX    0F
SYS$OPEN                        ********  GX    10
SYS$PARSE                       ********  GX    0F
SYS$QIO                         ********  GX    0F
SYS$QIOW                        ********  GX    0F
SYS$READEF                      ********  GX    0F
SYS$SCHDWK                      ********  GX    0F
SYS$SETEF                       ********  GX    0F
SYS$SETIMR                      ********  GX    0F
SYS$TRNLOG                      ********  GX    0F
SYS$ULWSET                      ********  GX    0F
SYS$WFLOR                       ********  GX    0F
SYS_ID_ADRS                     00000141 R      09
SYS_ID_LIST                     00002278 R      03
TERM_LIST                       00000572 R      02
TERM_LIST_LEN               =   0000000C
TIMBUF                          00001F96 R      03
TIME                        =   00000005
TIME_BUF                        000020BD R      03
TIME_DISPLAY                    0000259B R      0F
TIME_DSCR                       000020B5 R      03
TIME_LEN                    =   00000014
TIME_SOURCE                     00000000 R      02
TIME_STRING_LEN             =   0000000F
TITLE_BUF                       00001193 R      03
TITLE_BUF_LEN               =   00000400
TITLE_COUNT                     000021BC R      03
TITLE_DSCR                      0000118B R      03
TITLE_LEN_ARRAY                 000021D0 R      03
TITLE_START                     000021C0 R      03
TKN_DSCR                        00000495 R      03
TOP_CMD_LINE                    00000470 R      03
TOP_T                       =   00000077
TPA$K_COUNTO                =   00000008
TPA$K_LENGTHO               =   00000024
TPA$L_PARAM                 =   00000020
TPA$L_STRINGCNT             =   00000008
TPA$L_STRINGPTR             =   0000000C
TPA$L_TOKENCNT              =   00000010
TPA$L_TOKENPTR              =   00000014
TPA$M_ABBREV                =   00000002
TPA$_ALPHA                  =   000001EE
TPA$_ANY                    =   000001ED
TPA$_BLANK                  =   000001F2
TPA$_DECIMAL                =   000001F3
TPA$_DIGIT                  =   000001EF
TPA$_EOS                    =   000001F7
TPA$_EXIT                   =   FFFFFFFF
TPA$_FAIL                   =   FFFFFFFE
TPA$_FILESPEC               =   000001EA
TPA$_HEX                    =   000001F5
TPA$_IDENT                  =   000001EC
TPA$_KEYWORD                =   00000100
TPA$_LAMBDA                 =   000001F6
TPA$_MAXKEY                 =   000000DC
TPA$_OCTAL                  =   000001F4
TPA$_STRING                 =   000001F0
TPA$_SUBXPR                 =   000001F8
TPA$_SYMBOL                 =   000001F1
TPA$_UIC                    =   000001EB
TPARSE_BLK                      000004C7 R      03
TRANSITION_TIME_ADRS            0000046B R      09
TRAN_BUF                        000003A4 R      03
TRAN_DSCR                       0000039C R      03
TRM$M_TM_NOEDIT             =   00008000
TRM$_MODIFIERS              =   00000000
TRY_LOCK                        00003314 R      0F
TT$M_ESCAPE                 =   00000008
TT$M_NOBRDCST               =   00020000
TT$M_SCOPE                  =   00001000
TT$M_WRAP                   =   00000200
TT2$M_AVO                   =   08000000
TT2$M_BRDCSTMBX             =   00000010
TT2$M_DECCRT                =   20000000
TYPEAHD_BUF                     000004BD R      03
UCB$B_DEVTYPE               =   00000041
UCB$B_ERTCNT                =   00000080
```

```
UCB$L_PDT                       = 00000084
UCB$V_ONLINE                    = 00000004
UCB$W_STS                       = 00000064
UL_CORNER                       = 0000006C
UNACKED_ADRS                      00000309 R      09
UR_CORNER                       = 0000006B
VALUE                             0000054F R      03
VAR_BUF                           000003ED R      03
VAR_BUF_ADRS                      00000047 R      10
VAR_BUF_END                       000003F1 R      03
VAR_BUF_LEN                       000003F5 R      03
VERT_BAR                        = 00000078
VOTES_ADRS                        000003AD R      09
VT52_DWN_ARROW                  = 00000042
VT52_LFT_ARROW                  = 00000043
VT52_RT_ARROW                   = 00000044
VT52_UP_ARROW                   = 00000041
V_SCROLL                          0000045C R      03
V_SCROLL_LINE                     00000460 R      03
WARMCDRPS_ADRS                    00000369 R      09
WAS_LOCKED                        0000330A R      0F
WIDE                            = 00000084
YES_NO_STRING                     00000404 R      02
YES_NO_STRNG_LEN                = 00000003
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000360 (864.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	000005CE (1486.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
RWDATA	000022E4 (8932.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
$RMSNAM	00000010 (16.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CLUSTER_SIZE_ARRAY	00000046 (70.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
CLUSTER_BITMAP	00000018 (24.)	06 (6.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
CLUSTER_TABLE	00000118 (280.)	07 (7.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
CLUSTER_KEYTABLE	00000234 (564.)	08 (8.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
CLUSTER_DSCRS	00000923 (2339.)	09 (9.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
LOCAL_SIZE_ARRAY	00000019 (25.)	0A (10.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
LOCAL_BITMAP	00000008 (8.)	0B (11.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
LOCAL_TABLE	00000064 (100.)	0C (12.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
LOCAL_KEYTABLE	000000CC (204.)	0D (13.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
LOCAL_DSCRS	00000369 (873.)	0E (14.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
MAIN	00003333 (13107.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
SNAP_DATA	00001362 (4962.)	10 (16.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
_LIB$STATE$	00000652 (1618.)	11 (17.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
_LIB$KEY0$	00000098 (152.)	12 (18.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
_LIB$KEY1$	000001F5 (501.)	13 (19.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	10	00:00:00.07	00:00:00.57
Command processing	74	00:00:00.92	00:00:05.42
Pass 1	2671	00:02:27.22	00:07:54.26
Symbol table sort	0	00:00:07.28	00:00:27.10
Pass 2	1416	00:00:43.32	00:02:28.14
Symbol table output	5	00:00:01.04	00:00:03.65
Psect synopsis output	1	00:00:00.07	00:00:00.14
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	4180	00:03:19.92	00:10:59.28

The working set limit was 4100 pages.
851980 bytes (1665 pages) of virtual memory were used to buffer the intermediate code.
There were 240 pages of symbol table space allocated to hold 4034 non-local and 603 local symbols.
13210 source lines were read in Pass 1, producing 224 object records in Pass 2.
134 pages of virtual memory were used to define 114 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]PALIB.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	17
_255DUA28:[SYSLIB]STARLET.MLB;2	60
TOTALS (all libraries)	79

3970 GETS were required to define 79 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SHWCLSTR/OBJ=OBJ$:SHWCLSTR MSRC$:SHWCLSTR/UPDATE=(ENH$:SHWCLSTR)+EXECML$/LIB+SHRLIB$:PALIB/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SHWCLSTR
LIS

SHOWUSER
LIS

0059 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TERMDEFS
LIS

SUBMITMSG
LIS

TRANQUEUE
LIS

SWITCHTRM
LIS

SUBMIT
LIS